



**Calhoun: The NPS Institutional Archive**  
**DSpace Repository**

---

Theses and Dissertations

1. Thesis and Dissertation Collection, all items

---

1975

# An implementation of an iterative global flow analysis algorithm

Cowan, Jack William

Monterey, California. Naval Postgraduate School

---

<http://hdl.handle.net/10945/20894>

---

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

*Downloaded from NPS Archive: Calhoun*



Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

**Dudley Knox Library / Naval Postgraduate School**  
**411 Dyer Road / 1 University Circle**  
**Monterey, California USA 93943**

<http://www.nps.edu/library>

AN IMPLEMENTATION OF AN  
ITERATIVE GLOBAL  
FLOW ANALYSIS ALGORITHM.

Jack William Cowan

Library  
Naval Postgraduate School  
Monterey, California 93940

Gaylord  
CASE BINDER  
Syracuse, N. Y.  
Stockton, Calif.

# NAVAL POSTGRADUATE SCHOOL

Monterey, California



## THESIS

AN IMPLEMENTATION OF AN  
ITERATIVE GLOBAL  
FLOW ANALYSIS ALGORITHM

by

Jack William Cowan

March 1975

Thesis Advisor:

G. A. Kildall

Approved for public release; distribution unlimited.

T168325





Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) AN IMPLEMENTATION OF AN ITERATIVE GLOBAL FLOW ANALYSIS ALGORITHM		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis: March 1975
7. AUTHOR(s)  Jack W. Cowan		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (If different from Controlling Office) Naval Postgraduate School Monterey, California 93940		12. REPORT DATE March 1975
		13. NUMBER OF PAGES 290
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report)  Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Global Optimization, Program Flow Graph, Basic Block, Meet Operator, Code Synthesis Filter.		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Kildall has stated a general data flow analysis algorithm which has been applied to several forms of classical global program optimization. The algorithm operates upon the flow graph of a program, where the nodes correspond to basic blocks and the edges represent possible program control flows. In order to test the effectiveness of this algorithm, a general purpose optimizing module was written in XPL which analyzes		



Unclassified

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

ALGOL-E programs for constant computations, common subexpressions and simplifying formal identities. Various node selection algorithms were investigated with respect to the convergence rate of the algorithm.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)



An Implementation of an  
Iterative Global  
Flow Analysis Algorithm

by

Jack William Cowan  
Lieutenant(JG), United States Navy  
B.S., Naval Postgraduate School, 1974

Submitted in partial fulfillment of the  
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL  
March 1975





## ABSTRACT

Kildall has stated a general data flow analysis algorithm which has been applied to several forms of classical global program optimization. The algorithm operates upon the flow graph of a program, where the nodes correspond to basic blocks and the edges represent possible program control flows. In order to test the effectiveness of this algorithm, a general purpose optimizing module was written in XPL which analyzes ALGOL-E programs for constant computations, common subexpressions and simplifying formal identities. Various node selection algorithms were investigated with respect to the convergence rate of the algorithm.



## TABLE OF CONTENTS

I.	INTRODUCTION .....	10
	A. BACKGROUND .....	10
	B. DEFINITIONS .....	11
	C. A GLOBAL FLOW ANALYSIS ALGORITHM .....	15
II.	IMPLEMENTATION OF ALGORITHM Q .....	20
	A. CONCEPTUAL OVERVIEW .....	20
	1. Code Synthesis Filter Generator (CSFG) ..	22
	2. Code Synthesis Filter Emitters (CSFE) ...	27
	B. OPTIMIZING FUNCTIONS INCORPORATED IN THE CSF.	44
	1. Constant Propagation .....	44
	2. Simplifying Formal Identities .....	46
	3. Common Subexpression Elimination .....	46
	C. CODE SYNTHESIS FILTER (CSF) .....	47
	1. Program Flow .....	47
	2. Implementation Methodology .....	48
	a. Control Structures .....	48
	b. Meta-Execution .....	53
	c. Meet Operation .....	65
	d. Output Formats .....	72
III.	TESTING OF BLOCK SELECTION METHOD .....	87
IV.	CONCLUSIONS .....	104
	COMPUTER OUTPUT .....	106



TABLE OF CONTENTS (CON'T)

COMPUTER PROGRAM .....	214
BIBLIOGRAPHY .....	285
INITIAL DISTRIBUTION LIST .....	290



## LIST OF TABLES

I.	GRAMMAR FOR CSF GENERATION DESCRIPTION LANGUAGE ..	24
II.	CSF GENERATOR SPECIFICATION FOR ALGOL-E .....	28
III.	CSFG OUTPUT FOR ALGOL-E .....	31
IV.	ALGOL-E CODE INTERLISTED PROGRAM .....	45
V.	TABLE TRACE OUTPUT FROM CSF .....	75
VI.	DATA FOR STUDENT TEST # 1 .....	90
VII.	DATA FOR STUDENT TEST # 2 .....	92
VIII.	DATA FOR GENERALIZED STRUCTURE TEST .....	93
IX.	SUMMARY FOR STUDENT TEST # 1 .....	94
X.	SUMMARY FOR STUDENT TEST # 2 .....	95
XI.	SUMMARY FOR GENERALIZED STRUCTURE TEST .....	96
XII.	SUMMARY FOR ALL TESTS .....	97
XIII.	ANALYSIS OF CODE SIZE LESS THAN 200 WORDS .....	98
XIV.	ANALYSIS OF CODE SIZE FROM 200 TO 400 WORDS .....	99
XV.	ANALYSIS OF CODE SIZE FROM 400 TO 600 WORDS .....	100
XVI.	ANALYSIS OF CODE SIZE FROM 600 TO 800 WORDS .....	101
XVII.	ANALYSIS OF CODE SIZE GREATER THAN 800 WORDS .....	102





## LIST OF FIGURES

1.	AN ALGOL PROGRAM AND ITS ASSOCIATED FLOWGRAPH .....	12
2.	EXAMPLE OF FINAL POOL CONFIGURATION .....	19
3.	ORGANIZATION OF THE ALGORITHM Q IMPLEMENTATION ...	21
4.	INTERMEDIATE LANGUAGE OUTPUT FORMAT FROM CSFE .....	39
5.	OVERALL FLOW OF CSF .....	49
6.	OP_RANGE TO OP_CODES RELATIONSHIP .....	60
7.	OP_INDEX TO OP_INFO RELATIONSHIP .....	62
8.	SIMPLIFICATION LINK STRUCTURE .....	64
9.	RELATIONSHIP OF CODE SIZE AND BLOCKS PROCESSED ...	103



## ACKNOWLEDGEMENT

Without the patience and assistance of the Computer Operations staff this project would have been much less enjoyable. In particular, the efficient and reliable operating procedures of Edwin Donnellan, Mannas Anderson and Kristina Butler are deeply appreciated. A word of appreciation must also be extended to Katheryn Strutynski for her assistance with enumerable system problems. Finally, I am grateful to my wife, Patricia, for her many long and arduous hours keypunching and in obtaining needed reference materials. Without her willing and able assistance this project would never have been concluded.



## I. INTRODUCTION

### A. BACKGROUND

The increased use of high level language compilers has generated an interest in producing efficient machine language code from the high level source language. Generally, this interest has resulted in techniques for transforming the original source program to produce an equivalent form which is optimized with respect to some criteria.

Machine independent optimization algorithms can be divided into two broad subclasses - local and global. Basically, the difference is that global techniques take into account the overall flow of the program to be optimized while local methods do not. Specifically, local techniques require that the program to be optimized be partitioned into "basic blocks." These blocks are divisions of the program which have no transfers into or out of the block except for a transfer into an initial element and a transfer out of the last element. Each block is then optimized independent of the other existing blocks.

A number of techniques have evolved for analyzing global program structure, including recent work by Kildall [30], Hecht and Ullman [22], Allen [4], Kennedy [25], Graham and Wegman [20]. These techniques analyze global flow by representing the program as a directed graph with the nodes of the graph corresponding to basic blocks with the edges showing possible program control flows. If the ALGOL program of Figure 1 is analyzed for redundant calculations, local techniques would not detect the obvious redundancy of  $(C + B)$  in block D because this occurrence of the expression





resides in a different block than the previous calculation of  $(B + C)$ . On the other hand, global flow analysis would detect the redundant calculation since  $(B + C)$  has been computed on all paths to block D.

An iterative global flow analysis algorithm was developed by Kildall [28] that is capable of analyzing any program structure. The algorithm propagates data from block to block until the propagated data reaches a terminal state. This process may necessitate multiple passes through some blocks. Kildall has shown that the algorithm is finite and will succeed given any order for processing the basic blocks. However, using a simulation, Lukasczyk [34] has shown that the convergence rate of the algorithm can be affected by the order in which the "basic blocks" are processed. As one would expect, his simulation further indicates that as the number of nodes in a program flow graph increases, the method of block selection becomes more critical.

An implementation of the global flow analysis algorithm to be presented was originally done by Kildall [30]. Various improvements in the implementation were made, including more complete data structures for computing flow information. The results of Lukasczyk's simulation was then compared to the actual results from the implementation.

## B. DEFINITIONS

In order to examine the global flow analysis algorithm stated by Kildall [30], various basic notions need be presented.



```

IF X > Y THEN
BEGIN
  A := B + C;
  Y := X;
END
ELSE
  H := B + C;
R := C + B;

```

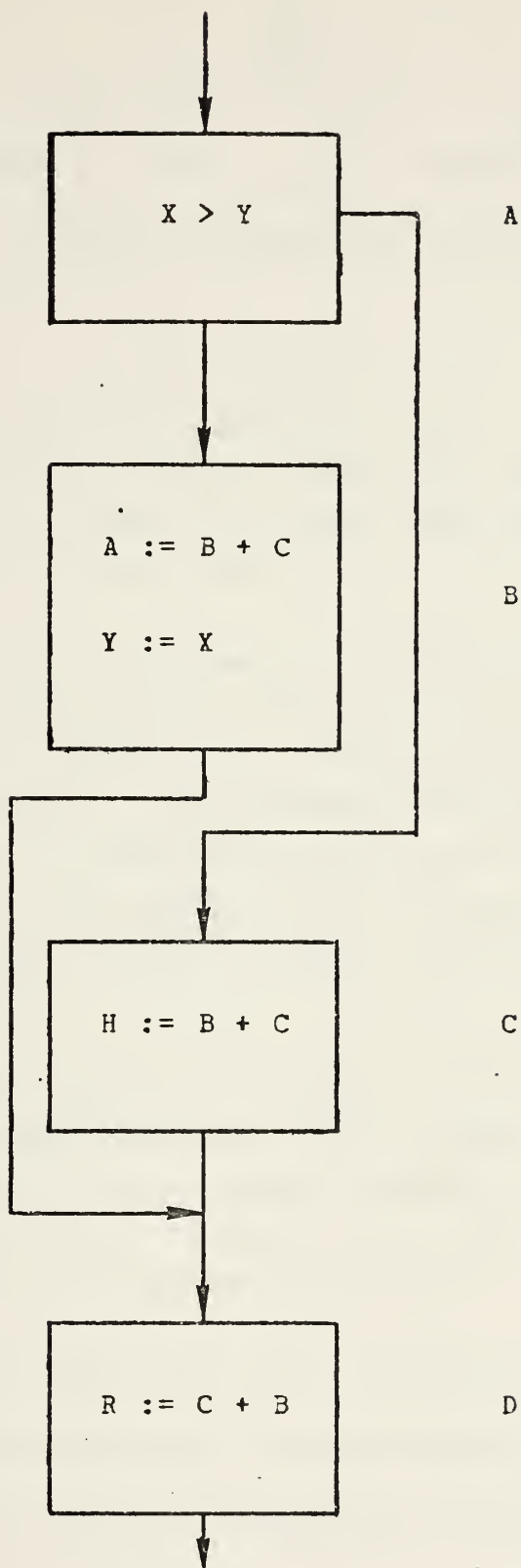


FIGURE 1

AN ALGOL PROGRAM AND ITS FLOWGRAPH



A directed graph  $G$  is a two-tuple  $(N, E)$ , where  $N$  represents the set of "nodes" and  $E$  is a subset of  $N \times N$  called "edges." Given an ordered pair of nodes  $(n_1, n_2)$  in  $E$ , an edge is said to "leave" node  $n_1$  and "enter" node  $n_2$ . Further,  $n_1$  is an "immediate predecessor" of  $n_2$ , and  $n_2$  is an "immediate successor" of  $n_1$ .

A program flow graph  $G$  is a three-tuple  $(N, E, R)$ , where  $N$  and  $E$  are as above and  $R$  is a non-empty subset of  $N$  containing the entry nodes of  $G$  such that there exists a path from an entry node to any node in  $N$ .

As stated previously, a program can be partitioned into a set of basic blocks. Each basic block is a program segment having no transfers entering or exiting except thru the initial and final elements. Basic blocks are represented by single nodes in the program flow graph and are hereafter referred to as blocks. The edges of the program flow graph represent the possible paths of control flow as directed by the program structure.

In the following discussion, note that for simplicity the various pools of information are always associated with the node "n." Where this rather informal association becomes unclear, particular effort will be made to distinguish pools with their associated nodes.

A current optimizing pool  $(P_c)$  is associated with a particular block  $n$  and represents the optimizing information available at the block, which is iteratively refined during the flow analysis.



An input pool ( $P_i$ ) is information which is to be used upon entry to a given block  $n$ . The optimizing function, defined below, uses this pool in its transformation. This information is identical to the current optimizing pool of a block  $n$ , and will be referred to as the input pool to increase the clarity of the global flow analysis algorithm which is stated below.

An output pool ( $P_o$ ) is optimizing information created by the application of the optimizing function to an input pool for a block. Upon exiting a block, this set is present and is used in performing the meet operator which is defined below.

A final optimized pool ( $P_f$ ) is the final optimizing information that is distinguished as the meet over all paths for a block  $n$ . This pool is produced by the global flow analysis algorithm for each block.

A meet operator is a binary operation denoted  $\wedge$ , on  $P \times P$  to  $P$ , where  $P$  is the set of all possible optimizing pools. The operator combines an output pool ( $P_o$ ) for a block  $n$  and a current optimized pool ( $P_c'$ ) of an immediate successor block  $n'$  to produce an input pool ( $P_i'$ ) for  $n'$ .

$$P_i' \leftarrow P_o \wedge P_c'$$

The following properties must hold for all  $a$ ,  $b$  and  $c$  in  $P$ :

- 1)  $a \wedge a = a$  (idempotent)
- 2)  $a \wedge b = b \wedge a$  (commutative)
- 3)  $a \wedge (b \wedge c) = (a \wedge b) \wedge c$  (associative)





The meet operation results in a partial ordering upon the optimization elements:  $a \leq b$  if and only if  $a \wedge b = a$ .

An optimizing function maps an input pool to an output pool and must have the homomorphism property:

$$f(n, a \wedge b) = f(n, a) \wedge f(n, b)$$

for all nodes  $n$  in  $N$  and input pools  $a$  and  $b$  in  $P$ .  $P$  is assumed to contain an identity element such that

$$1 \wedge a = a \text{ for all } a \text{ in } P.$$

### C. A GLOBAL FLOW ANALYSIS ALGORITHM

Although a basic familiarization with Kildall's flow analysis algorithm [28] is assumed, some fundamental notions are presented below for completeness.

The global flow analysis algorithm, Algorithm Q, of Kildall was developed for use in the compile time optimization of object code. Kildall [30] has pointed out that many techniques have been developed to optimize straight line code while others take into account program flow branching. The Algorithm Q was designed to permit extensions of straight line optimization techniques to the global case.

The Algorithm Q is iterative and may be used on any program flow graph. As stated previously, the algorithm propagates data from block to block until the propagated data reaches a final state for a given block. This final state is the "meet over all paths," called the MOP solution, for a specified block and provides information with which the block can be optimized.



Informally the Algorithm Q can be stated as:

Select a node from the list of nodes to be processed, its current pool will be used as input.

Apply the optimizing function to the input pool and produce an output pool.

For each immediate successor, apply the meet operation on the output pool and the successor's current pool. If the result is strictly smaller than the successor's current pool, then reset the current pool to this result and add the successor to the nodes to be processed list.

A particular form of the Algorithm Q which is appropriate for machine implementation is stated below.

Q1 [initialize]	initialize the investigation list L to the entry node $L \leftarrow \{n\}$ and set the current pool of n to empty. $P_c \leftarrow 0$
Q2 [terminate?]	if L is empty then halt, otherwise
Q3 [select a node]	let n be an element of L, and set $L \leftarrow L - \{n\}$ .
Q4 [apply function]	apply the optimizing function to the $P_c$ of n producing



output information  $P_o$  for  
the node  $n$ .

Q5 [enter nodes]

for each  $n'$  which  
is an immediate successor of  
 $n$ , form  $P_i' \leftarrow P_o \wedge P_c'$

where  $P_i'$  and  $P_c'$  correspond

to node  $n'$ . If  $P_i' \leq P_c'$

and  $P_i' \neq P_c'$  then set

$P_c' \leftarrow P_i'$

and

$L \leftarrow L \cup \{n'\}$ .

Q6 [loop]

go to Q3.

As an example of the Algorithm Q, consider common subexpression analysis. The pools are partitions of available expressions which are in the same class if they are known to have the same value. For example, the result of analyzing the statements

```
A := 2;
B := 3;
C := A + B;
```

could result in the pool

$\{A, 2 \mid B, 3 \mid C, A + B, B + A, 2 + 3, 3 + 2, 5\}$

where " $\mid$ " is the delimiter for the different equivalence classes. The meet operator is defined as a simple intersection operator upon the partition elements, and the optimizing function builds new equivalence classes from existing equivalence classes, based upon operations at a particular node.



Upon termination of the Algorithm Q, the final optimization pool ( $P_f$ ) for each node can be found by selecting the step in which each node was last processed. The current pool ( $P_c$ ) at this step will be the final optimizing pool ( $P_f$ ) which can be used in a subsequent pass through the block to perform the actual program transformations.

Consider the ALGOL program of Figure 1. The basic blocks are A, B, C and D with the entry node A and corresponding initial optimizing pool set to empty. Figure 2 gives the  $P_f$  for the program of Figure 1. Only a single pass of each block was required since there are no loops in the program flow graph. In general, the algorithm requires multiple passes on some blocks to produce the final optimizing pool. A complete and formal discussion of the global flow analysis algorithm by Kildall can be found in [28, 29, 30].





NODE	$P_f$
A	null
B	$\{x \mid y$ $\mid x > y\}$
C	$\{a, b+c,$ $c+b \mid$ $y, x\}$
D	$\{a, h,$ $b + c,$ $c + d \mid$ $y, x\}$

FIGURE 2  
EXAMPLE OF FINAL POOL CONFIGURATION



## II. IMPLEMENTATION OF ALGORITHM Q

### A. CONCEPTUAL OVERVIEW

The original implementation of Algorithm Q was given by Kildall [30], and this project is basically an extension of Kildall's original efforts.

Implementation of the global flow analysis algorithm consists of three separate modules written in XPL [37]. The first module, the "Code Synthesis Filter Generator," or CSFG, creates tables which are used by the other two modules. A set of procedures known as the "Code Synthesis Filter Emitters," CSFE, constitute the second module which is used with the XPL skeletal compiler generator. The CSFE produces the intermediate language which is to be optimized. A particular compiler was implemented using the CSFE for a language called ALGOL-E, which is described elsewhere [32]. The third module implements the global flow analysis algorithm itself, and is called the "Code Synthesis Filter," or CSF.

Communication between the CSFE and CSF is through an intermediate language which represents a program in Polish form. The CSF reads the intermediate language from the compiler and flags optimization information, without actually performing any program transformations. Figure 3 outlines the flow of the present three module system.

The next two sections contain a description of the CSFG and CSFE as implemented by Kildall [30]. They are included here to ensure the completeness of the later CSF description and to simplify the discussion of the CSF module.



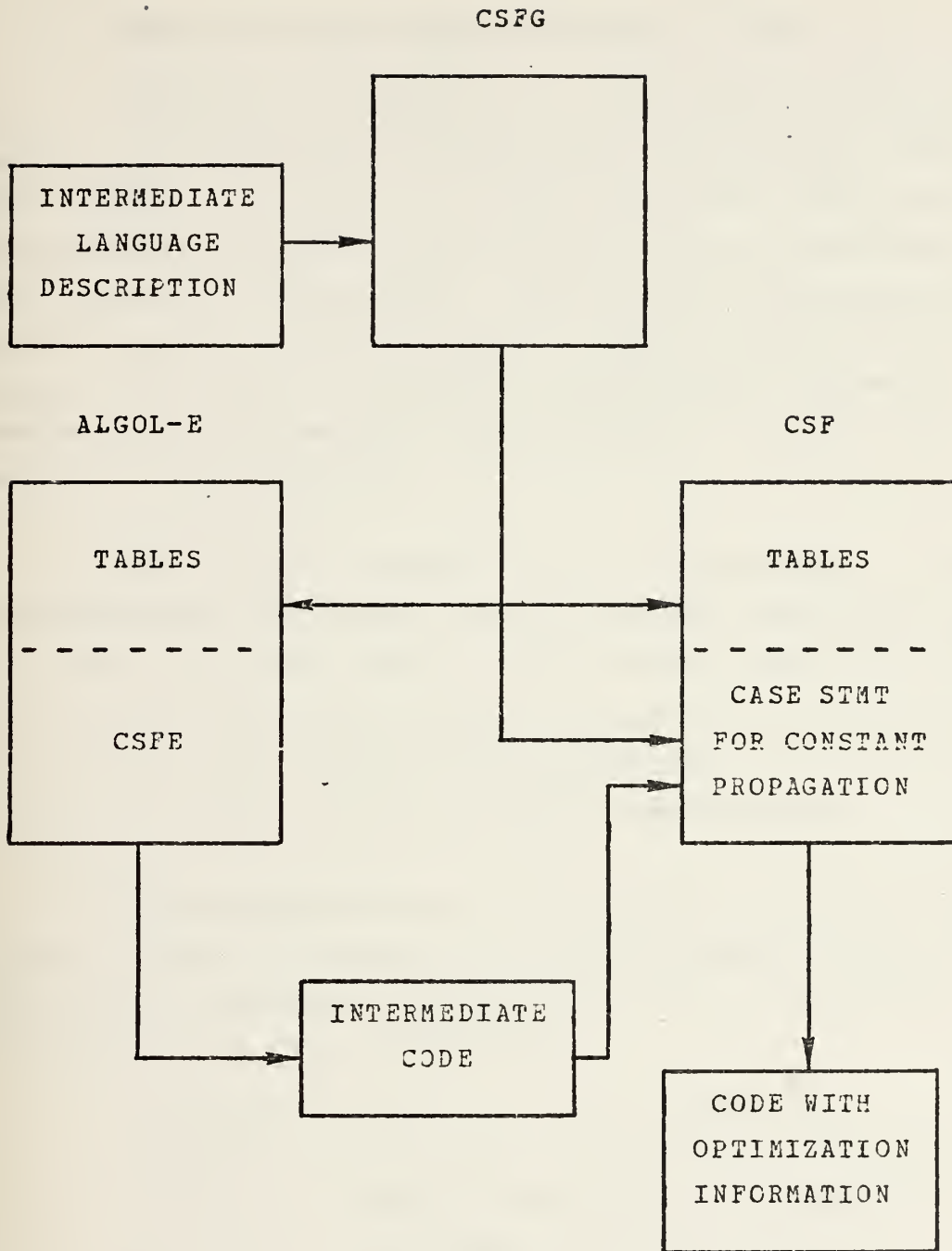


FIGURE 3  
ORGANIZATION OF THE ALGORITHM Q IMPLEMENTATION



## 1. Code Synthesis Filter Generator (CSFG)

Primarily, the function of the CSFG is to provide a method for generating a customized intermediate language. The output of CSFG is a set of tables which define an intermediate language, along with an XPL case statement used for constant propagation. The CSF and CSFE are driven by these tables. The intermediate language takes the form of a string of Polish operators and operands, augmented with control flow information, which assumes a stack model for semantic interpretation.

The CSFG accepts an intermediate language specification determined by the grammar specified in TABLE I. There are three parts to the language specification:

- 1) <TYPE SPECIFICATION>
- 2) <OPCODE SPECIFICATION>
- 3) <SIMPLIFICATION SPECIFICATION>.

<TYPE SPECIFICATION> is primarily a list of variable types. Those followed by an asterisk denote data elements that occupy "regions" upon execution (i.e., an array). The following types are predeclared:

- 1) DEFAULT - type cannot be categorized,
- 2) LOC - location variable (i.e., address),
- 3) ENT - program entry point or branch  
operation destination point,
- 4) XIT - program branch point.

An XIT (exit) is the operand of a branch instruction while the ENT (entry) type variable is placed in the Polish sequence at the destination of a branch.





<OPCODE SPECIFICATION> gives the descriptive characteristics for each operator allowed in the Polish string. The following characteristics are generated for each OPCODE:

- 1) operator name
- 2) special attributes of the operator
- 3) number of operands necessary  
to perform the operation
- 4) type of each operand
- 5) the number of operands resulting  
from an operator
- 6) type of the resulting operands  
of an operator.

Special operators (REFER, PASS and TOGGLE) are predefined and require a LOC type argument with no resulting operands. These operators are used for special purposes discussed in Section II.A.2.

Special operator attributes aid in the optimization process by indicating operators which affect program flow or which have special effects on the execution stack. Specifically, these attributes are:

- 1) LOAD - results in a fetch from memory.  
The operand must be of type LOC  
(i.e., an address from which to fetch).
- 2) STORE - results in a store of the second  
operand into the address specified by the first  
operand. The first operand must be type LOC.  
The stack is cleared of both operands upon  
completion.



TABLE I

GRAMMAR FOR CSF GENERATION DESCRIPTION LANGUAGE

```

<CSF SPEC> ::= <TYPE SPEC> <OPCODE SPEC>
              | <TYPE SPEC> <OPCODE SPEC>
              | <SIMP SPEC>

<TYPE SPEC> ::= <TYPE HEAD> <TYPE ELEMENT> ;

<TYPE HEAD> ::= <TYPES>
              | <TYPE HEAD> <TYPE ELEMENT> ,

<TYPES> ::= TYPES

<TYPE ELEMENT> ::= <IDENTIFIER>
                  <IDENTIFIER> *

<OPCODE SPEC> ::= <OPCODE HEAD> <OPCODE RULE> ;

<OPCODE HEAD> ::= <OPCODES>
                 | <OPCODE HEAD> <OPCODE RULE>

<OPCODES> ::= OPCODES

<OPCODE RULE> ::= <OPCODE FORMAT>
                 | <OPCODE FORMAT> <TYPE>
                 | <OPCODE FORMAT> *

<OPCODE FORMAT> ::= <OPCODE PARMS> = >

<OPCODE PARMS> ::= <OPCODE PARM HEAD>
                  | <OPCODE PARMS> <TYPE>
                  | <OPCODE PARMS> *

<OPCODE PARM HEAD> ::= <IDENTIFIER>
                     | <IDENTIFIER>
                     | (<OPCODE TYPE>)

```



TABLE I (CON'T)

```

<OPCODE TYPE> ::= LOAD
                | STORE
                | STORET
                | DELETE
                | DUPLIC
                | CONV
                | COMM
                | EXCH
                | COND
                | UCOND
                | GOSUB
                | RETSUB

<SIMP SPEC> ::= <SIMP HEAD> <SIMP RULE> ;

<SIMP HEAD> ::= <SIMPLIFICATIONS>
                | <SIMP HEAD> <SIMP RULE> ,

<SIMPLIFICATIONS> ::= SIMPLIFICATIONS

<SIMP RULE> ::= <SIMP FORMAT> <IDENTIFIER>
                | <SIMP FORMAT> <STRING>

<SIMP FORMAT> ::= <SIMP LEFT PART> = >

<SIMP LEFT PART> ::= <SIMP RULE HEAD> <IDENTIFIER>
                    | <SIMP RULE HEAD> <STRING>
                    | <SIMP RULE HEAD> *

<SIMP RULE HEAD> ::= <OPCODE>
                    | <SIMP RULE HEAD> <IDENTIFIER>
                    | <SIMP RULE HEAD> <STRING>
                    | <SIMP RULE HEAD> *

```



- 3) STORET - the same as STORE; however, the second operand is retained on the stack for future use.
- 4) DELETE - removes the top entry from the stack (i.e., normally an operand).
- 5) DUPLIC - duplicates the top entry on the stack.
- 6) CONV - type conversion operators.
- 7) COMM - operator is commutative in all its operands.
- 8) EXCH - exchanges top two entries on the stack.
- 9) COND - conditional branch operator.
- 10) UCOND - unconditional branch operator.
- 11) GOSUB - branch to a procedure.
- 12) RETSUB - return branch from a procedure to the calling point.

<SIMPLIFICATION SPECIFICATION> defines acceptable transformations which may be performed on the operators. Each transformation is of the form

- 1) operator
- 2) operands
- 3) result operands.

For example:

ADD            "0"            A    =>    A

indicates that the ADD operator would result in the operand A if one operand is zero, where A represents any arbitrary expression.





TABLE II outlines the particular intermediate language description used for the ALGOL-E compiler. TABLE III provides the complete output from the CSFG for this description.

## 2. Code Synthesis Filter Emitters (CSFE)

The CSFE is a set of XPL procedures that is included as an integral part of the XPL skeletal compiler for any particular compiler generation. The CSFE uses the tables produced by the CSFG to generate the intermediate language which is compatible with the optimizer, but tailored to the specific source language and compiler being developed.

As discussed above, the intermediate language is a Polish sequence consisting of a string of operators, operands, and branch point references. A table of referenced constants is also included in the Polish sequence. Figure 4 shows the format of the CSFE output. Individual fields are:

- 1) OFFSET(8b) - used for diagnostic reporting.
- 2) OPCODE/TYPE(8b) - designates the opcodes if the entry is an operator, or designates the type of operand if the entry is an operand.
- 3) C(1b) - if set, indicates a reference to the constant tables; if not, reference is to an address.
- 4) ADDR(15b) - address field of operands in the intermediate language.



## TABLE II

### A SAMPLE CSF GENERATOR SPECIFICATION FOR ALGOL-E

```

/* $OUTPUT PRINTED TABLES */
/* $PUNCH TABLES */
TYPES INT,REAL,BOOL,IARRAY*,RARRAY*;
OPCODES
TRU (CONV) REAL => INT          /* TRUNCATE TAKES */
                                /* REAL TO INTEGER */
RND (CONV) REAL => INT          /* RND IS THE */
                                /* ROUND OPERATOR */
FLT (CONV) INT => REAL,         /* FLT IS THE */
                                /* FLOAT FUNCTION */
ADD (COMM) INT INT => INT,      /* INTEGER ADDITION */
                                /* (COMMUTATIVE) */
RADD (COMM) REAL REAL => REAL, /* REAL ADDITION */
SUB INT INT => INT,            /* INTEGER */
                                /* SUBTRACTION */
RSUB REAL REAL => REAL,        /* REAL SUBTRACTION */
MUL (COMM) INT INT => INT,      /* INTEGER */
                                /* MULTIPLICATION */
RMUL (COMM) REAL REAL => REAL, /* REAL SUBTRACTION */
DIV INT INT => INT,            /* INTEGER DIVISION */
RDIV REAL REAL => REAL,        /* REAL DIVISION */
EXP INT INT => INT,            /* INTEGER */
                                /* EXPONENTIATION */
RIXP REAL INT => REAL,         /* REAL TO INTEGER */
                                /* EXPONENT */
IRXP INT REAL => REAL,         /* INTEGER TO REAL */
                                /* EXPONENT */
RRXP REAL REAL => REAL,        /* REAL TO REAL */
                                /* EXPONENT */

```



TABLE II (CON'T)

LSS * * => BOOL,	/* LESS THAN TEST */
LEQ * * => BOOL,	/* LESS THAN OR */
	/* EQUAL TEST */
EQL (COMM) * * => BOOL,	/* EQUAL TO TEST */
NEQ * * => BOOL,	/* NOT EQUAL TO TEST */
GEQ * * => BOOL,	/* GREATER THAN OR */
	/* EQUAL TO */
GTR * * => BOOL,	/* GREATER THAN TEST */
INEG INT => INT,	/* INTEGER NEGATION */
RNEG REAL => REAL,	/* REAL NEGATION */
NOT BOOL => BOOL,	/* BOOLEAN NOT */
AND (COMM) BOOL BOOL => BOOL,	/* BOOLEAN AND */
BOR (COMM) BOOL BOOL => BOOL,	/* BOOLEAN OR */
LOD (LOAD) LOC => * ,	/* LOAD FROM ADDRESS */
STO (STORET) LOC * => * ,	/* STORE AND RETAIN */
	/* VALUE */
STD (STORE) LOC * => ,	/* STORE AND REMOVE */
	/* VALUE */
DEL (DELETE) * => ,	/* DELETE TOP */
	/* ELEMENT */
DUP (DUPLIC) => * ,	/* DUPLICATE */
	/* ELEMENT */
XCH (EXCH) => ,	/* EXCHANGE LAST */
	/* ELEMENTS */
HLT => ,	/* PROGRAM HALT */
BRS (UCOND) XIT => ,	/* UNCONDITIONAL */
	/* BRANCH */
BSC (COND) BOOL XIT => ,	/* CONDITIONAL */
	/* BRANCH */
NOP => ,	/* NO OPERATION */



TABLE II (CON'T)

PRO (GOSUB) XIT => ,	/* PROCEDURE */
	/* TRANSFER */
RTN (RETSUB) => ,	/* RETURN FROM */
	/* PROCEDURE */
GET INT => LOC,	/* GET STORAGE */
RET LOC => ,	/* RETURN STORAGE */
RRD => REAL,	/* REAL READ */
IRD => INT,	/* INTEGER READ */
BRD => BOOL,	/* BOOLEAN READ */
WRVx* => ,	/* WRITE VARIABLE */
DMP => ,	/* DUMP BUFFER */
TAB INT => ,	/* TABULATE */
SUP * => ;	
SIMPLIFICATIONS	
ADD "0" A => A,	/* 0+A=A+0=A */
RADD "0" A => A,	
MUL "0" A => "0",	/* 0*A=A*0=0 */
RMUL "0" A => "0",	
MUL "1" A => A,	/* 1*A=A*1=A */
RMUL "1" A => A,	
SUB A A => "0",	/* A-A= */
RSUB A A => "0",	
DIV "0" A => "0" ,	/* 0/A = 0 */
RDIV "0" A => "0" ,	/* 0/A = 0 */
EXP A "0" => "1" ,	/* A**0 = 1 */
RIXP A "0" => "1" ,	
IRXP A "0" => "1" ,	
RRXP A "0" => "1" ,	
EXP A "1" => A ,	/* A ** 1 = A */
RIXP A "1" => A,	/* A ** 1 = A */
IRXP A "1" => A,	
RRXP A "1" => A,	





# TABLE III

## CSFG OUTPUT FOR ALGOL-E

```

DECLARE(NULL, DEFAULT, LOC, ENT, XIT, INT, REAL, BOOL, IARRA
Y, RARRAY) BIT(8);
DECLARE NTYPES LITERALLY'9';
DECLARE(REFER, TOGGLE, PASS, TRU, RND, FLT, ADD, RADD, SUB,
RSUB, MUL, RMUL, DIV, RDIV, EXP, RIXP, IRXP, RXP, LSS, LEQ,
EQL, NEQ, GEQ, GTR, INEG, RNEG, NTO, AND, BOR, LOD, STO, ST
D, DEL, DUP, XCH, HLT, BRS, BSC, NOP, PRO, RTN, GET, RET, RR
D, IRD, BRD, WRV, DMP, TAB, SUP) BIT(8);
DECLARE NOPCODES LITERALLY'50';
DECLARE OPCODES CHARACTER INITIAL(
'NULLDEFAULTLOCENTXITINTREALBOOLIARRAYRARRAYREFERTOGGLEPASST
RURNDFLTADDRADDSUBRSUBMULRMULDIVRDIVEXPRIXPIRXPPRRXPLSSLEQE
QLNEQGEQGTRINEGRNEGNOTANDBORLODSTOSTDDELDUPXCHHLTBRSBSCNOPPR
ORTNGETRETRRDIRDBRDWRVDMPTABSUP');
DECLARE OPRANGE(60) BIT(8) INITIAL(0, 4, 11, 14, 17, 20, 23,
27, 31, 37, 43, 48, 54, 58, 61, 64, 67, 70, 74, 77, 81, 84,
88, 91, 95, 98, 102, 106, 110, 113, 116, 119, 122, 125, 128,
132, 136, 139, 142, 145, 148, 151, 154, 157, 160, 163, 166,
169, 172, 175, 178, 181, 184, 187, 190, 193, 196, 199, 202,
205, 205);
DECLARE OP_TYPE(59) BIT(8) INITIAL(0, 0, 0, 0, 0, 0, 0, 0,
1, 1, 0, 0, 0, 7, 7, 7, 8, 8, 0, 0, 8, 8, 0, 0, 0, 0, 0, 0,
0, 0, 8, 0, 0, 0, 0, 0, 0, 8, 8, 2, 4, 3, 5, 6, 9, 0, 11,
10, 0, 12, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0);
DECLARE OP_DEGL(59) BIT(8) INITIAL(0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 1, 1, 1, 2, 2, 1, 2, 2, 1, 0, 0, 0, 1, 2,
0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1);

```



TABLE III (CON'T)

```

DECLARE OP_DEGR(59) BIT(8) INITIAL(0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0);

DECLARE OP_INDEX(60) BIT(8) INITIAL(0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 3, 5, 7, 10, 13, 16, 19, 22, 25, 28, 31,
34, 37, 40, 43, 46, 49, 52, 55, 58, 61, 63, 65, 67, 70, 73,
75, 78, 80, 81, 82, 82, 82, 83, 85, 85, 86, 86, 88, 89, 90,
91, 92, 93, 93, 94, 95);

DECLARE OP_INFO(170) BIT(16) INITIAL(0, 6, 5, 6, 5, 5, 6,
5, 5, 5, 6, 6, 6, 5, 5, 5, 6, 6, 6, 5, 5, 5, 6, 6, 6, 5, 5,
5, 6, 6, 6, 5, 5, 5, 6, 5, 6, 5, 6, 6, 6, 6, 6, 1, 1, 7, 1,
1, 7, 1, 1, 7, 1, 1, 7, 1, 18 7, 1, 1, 7, 5, 5, 6, 6, 7, 7,
7, 7, 7, 7, 7, 7, 2, 1, 2, 1, 1, 2, 1, 1, 1, 4, 7, 4, 4, 5,
2, 2, 6, 5, 7, 1, 5, 1, 0, 3, 2, 2, 0, 3, 2, 2, 111, 3, 2,
3, 115, 3, 2, 3, 0, 5, 2, 2, 0, 5, 2, 2, 0, 2, 2, 3, 0, 2,
2, 3, 0, 3, 2, 3, 0, 3, 2, 3, 151, 2, 3, 5, 155, 2, 3, 5,
159, 2, 3, 5, 163, 2, 3, 5, 0, 2, 5, 2, 0, 2, 5, 2, 0, 2, 5,
2, 167, 2, 5, 2, 0, 2, 5, 2);

DECLARE SIMP_INDEX(59) BIT(8) INITIAL(0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 95, 99, 119, 123, 103, 107, 127,
131 135, 139, 143, 147, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 08 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);

DECLARE OP_STR(2) CHARACTER INITIAL('', '0', '1');

**** *
DECLARE(NULL, DEFAULT, LOC, ENT, XIT, INT, REAL, BOOL, IARRA
Y, RARRAY) BIT(8);

DECLARE NTYPES LITERALLY'9';

```



TABLE III (CON'T)

```

DECLARE(REFER, TOGGLE, PASS, TRU, RND, FLT, ADD, RADD, SUB,
RSUB, MUL, RMUL, DIV, RDIV, EXP, RIXP, IRXP, RXP, LSS, LEQ,
EQL, NEQ, GEQ, GTR, INEG, RNEG, NOT, AND, BOR, LOD, STO, ST
D, DEL, DUP, XCH, HLT, BRS, BSC, NOP, PRO, RTN, GET, RET, RR
D, IRD, BRD, WRV, DMP, TAB, SUP) BIT(8);
DECLARE NOPCODES LITERALLY'50';
DECLARE OPCODES CHARACTER INITIAL(
'NULLDEFAULTLOCENTXITINTREALBOOOLIAARRAYRARRAYREFERTOGGLEPASST
RURNDFLTADDRADDSUBRSUBMULRMULDIVRDXPRIXPIRXPRXPLSSLEQEQ
NEQGEQGTRINEGRNEGNOTANDBORLODSTOSTDDELDPXCHHLTBRSBSCNOPPROR
TNGETRETRRDIRDBRDWRVDMPTABSUP');
DECLARE OPRANGE(60) BIT(8) INITIAL(0, 4, 11, 14, 17, 20,
23, 27, 31, 37, 43, 48, 54, 58, 61, 64, 67, 70, 74, 77, 81,
84, 88, 91, 95, 98, 102, 106, 110, 113, 116, 119, 122, 125,
128, 132, 136, 139, 142, 145, 148, 151, 154, 157, 160, 163,
166, 169, 172, 175, 178, 181, 184, 187, 190, 193, 196, 199,
202, 205, 208);
DECLARE OP_TYPE(59) BIT(8) INITIAL(0, 0, 0, 0, 0, 0, 0, 0,
1, 1, 0, 0, 0, 7, 7, 7, 8, 8, 0, 0, 8, 8, 0, 0, 0, 0, 0, 0,
0, 0, 8, 0, 0, 0, 0, 0, 0, 8, 8, 2, 4, 3, 5, 6, 9, 0, 11,
10, 0, 12, 13, 08 0, 0, 0, 0, 0, 0, 0, 0);
DECLARE OP_DEGL(59) BIT(8) INITIAL(0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 1, 1, 1, 2, 2, 1, 2, 2, 1, 0, 0, 0, 1, 2,
0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1);
DECLARE OP_DEGR(59) BIT(8) INITIAL(0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0);

```



TABLE III (CON'T)

```

DECLARE OP_INDEX(60) BIT(8) INITIAL(0, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 3, 5, 7, 10, 13, 16, 19, 22, 25, 28, 31,
34, 37, 40, 43, 46, 49, 52, 55, 58, 61, 63, 65, 67, 70, 73,
75, 78, 80, 81, 82, 82, 83, 85, 85, 86, 86, 88, 89, 90,
91, 92, 93, 94, 95);

```

```

DECLARE OP_INFO(170) BIT(16) INITIAL(0, 6, 5, 6, 5, 5, 6,
5, 5, 5, 6, 6, 6, 5, 5, 5, 6, 6, 6, 5, 5, 5, 6, 6, 6, 5, 5,
5, 6, 6, 6, 5, 5, 5, 6, 5, 6, 5, 6, 6, 6, 6, 6, 1, 1, 7, 1,
1, 7, 1, 1, 7, 1, 1, 7, 1, 1, 7, 1, 1, 7, 5, 5, 6, 6, 7, 7,
7, 7, 7, 7, 7, 7, 2, 1, 2, 1, 1, 2, 1, 1, 1, 4, 7, 4, 4, 5,
2, 2, 6, 5, 7, 1, 5, 1, 0, 3, 2, 2, 0, 3, 2, 2, 111, 3, 2,
3, 115, 3, 2, 3, 0, 5, 2, 2, 0, 5, 2, 2, 0, 2, 2, 3, 0, 2,
2, 3, 0, 3, 2, 3, 0, 3, 2, 3, 151, 2, 3, 5, 155, 2, 3, 5,
159, 2, 3, 5, 163, 2, 3, 5, 0, 2, 5, 2, 0, 2, 5, 2, 0, 2, 5,
2, 167, 2, 5, 2, 0, 2, 5, 2);

```

```

DECLARE SIMP_INDEX(59) BIT(8) INITIAL(0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 95, 99, 119, 123, 103, 107, 127,
131, 135, 139, 143, 147, 0, 0, 0, 08 08 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
DECLARE OP_STR(2) CHARACTER INITIAL(' ', '0', '1');

```

```

/* CASE 0: NULL */

```

```

;

```

```

/* CASE 1: DEFAULT => (TYPE) */

```

```

;

```

```

/* CASE 2: LOC => (TYPE) */

```

```

;

```

```

/* CASE 3: ENT => (TYPE) */

```

```

;

```

```

/* CASE 4: XIT => (TYPE) */

```

```

;

```

```

/* CASE 5: INT => (TYPE) */

```

```

;

```





TABLE III (CON'T)

```

/* CASE 6: REAL => (TYPE) */
;
/* CASE 7: BOOL => (TYPE) */
;
/* CASE 8: IARRAY => (TYPE) */
;
/* CASE 9: RARRAY => (TYPE) */
;
/* CASE 9: RARRAY => (TYPE) */
;
/* CASE 10: REFER => (OPCODE) */
;
/* CASE 11: TOGGLE => (OPCODE) */
;
/* CASE 12: PASS => (OPCODE) */
;
/* CASE 13: TRU REAL => INT (OPCODE) */
;
/* CASE 14: RND REAL => INT (OPCODE) */
;
/* CASE 15: FLT INT => REAL (OPCODE) */
;
/* CASE 16: ADD INT INT => INT (OPCODE) */
;
/* CASE 17: RADD REAL REAL => REAL (OPCODE) */
;
/* CASE 18: SUB INT INT => INT (OPCODE) */
;
/* CASE 19: RSUB REAL REAL => REAL (OPCODE) */
;
/* CASE 20: MUL INT INT => INT (OPCODE) */
;
/* CASE 21: RMUL REAL REAL => REAL (OPCODE) */

```



# TABLE III (CON'T)

```

;
/* CASE 22: DIV INT INT => INT (OPCODE) */
;
/* CASE 23: RDIV REAL REAL => REAL (OPCODE) */
;
/* CASE 24: EXP INT INT => INT (OPCODE) */
;
/* CASE 25: RIXP REAL INT => REAL (OPCODE) */
;
/* CASE 26: IRXP INT REAL => REAL (OPCODE) */
;
/* CASE 27: RRXp REAL REAL => REAL (OPCODE) */
;
/* CASE 28: LSS DEFAULT DEFAULT => BOOL (OPCODE) */
;
/* CASE 29: LEQ DEFAULT DEFAULT => BOOL (OPCODE) */
;
/* CASE 30: EQL DEFAULT DEFAULT => BOOL (OPCODE) */
;
/* CASE 31: NEQ DEFAULT DEFAULT => BOOL (OPCODE) */
;
/* CASE 32: GEQ DEFAULT DEFAULT => BOOL (OPCODE) */
;
/* CASE 33: GTR DEFAULT DEFAULT => BOOL (OPCODE) */
;
/* CASE 34: INEG INT => INT (OPCODE) */
;
/* CASE 35: RNEG REAL => REAL (OPCODE) */
;
/* CASE 36: NOT BOOL => BOOL (OPCODE) */
;
/* CASE 37: AND BOOL BOOL => BOOL (OPCODE) */

```



# TABLE III (CON'T)

```

;
/* CASE 38: BOR BOOL BOOL => BOOL (OPCODE */
;
/* CASE 39: LOD LOC => DEFAULT (OPCODE */
;
/* CASE 40: STO LOC DEFAULT => DEFAULT (OPCODE) */
;
/* CASE 41: STD LOC DEFAULT => (OPCODE) */
;
/* CASE 42: DEL DEFAULT => (OPCODE) */
;
/* CASE 43: DUP => DEFAULT (OPCODE) */
;
/* CASE 44: XCH => (OPCODE) */
;
/* CASE 45: HLT => (OPCODE) */
;
/* CASE 46: BRS XIT => (OPCODE) */
;
/* CASE 47: BSC BOOL XIT => (OPCODE) */
;
/* CASE 48: NOP => (OPCODR) */
;
/* CASE 49: PRO XIT => (OPCODE) */
;
/* CASE 50: RTN => (OPCODE) */
;
/* CASE 51: GET INT => LOC (OPCODE) */
;
/* CASE 52: RET LOC => (OPCODE) */
;
/* CASE 53: RRD => REAL (OPCODE) */

```



TABLE III (CON'T)

```
;  
/* CASE 54: IRD => INT (OPCODE) */  
;  
/* CASE 55: BRD => BOOL (OPCODE) */  
;  
/* CASE 56: WRV DEFAULT => (OPCODE) */  
;  
/* CASE 57: DMP => (OPCODE) */  
;  
/* CASE 58: TAB INT => (OPCODE) */  
;  
/* CASE 59: SUP DEFAULT => (OPCODE) */  
;
```





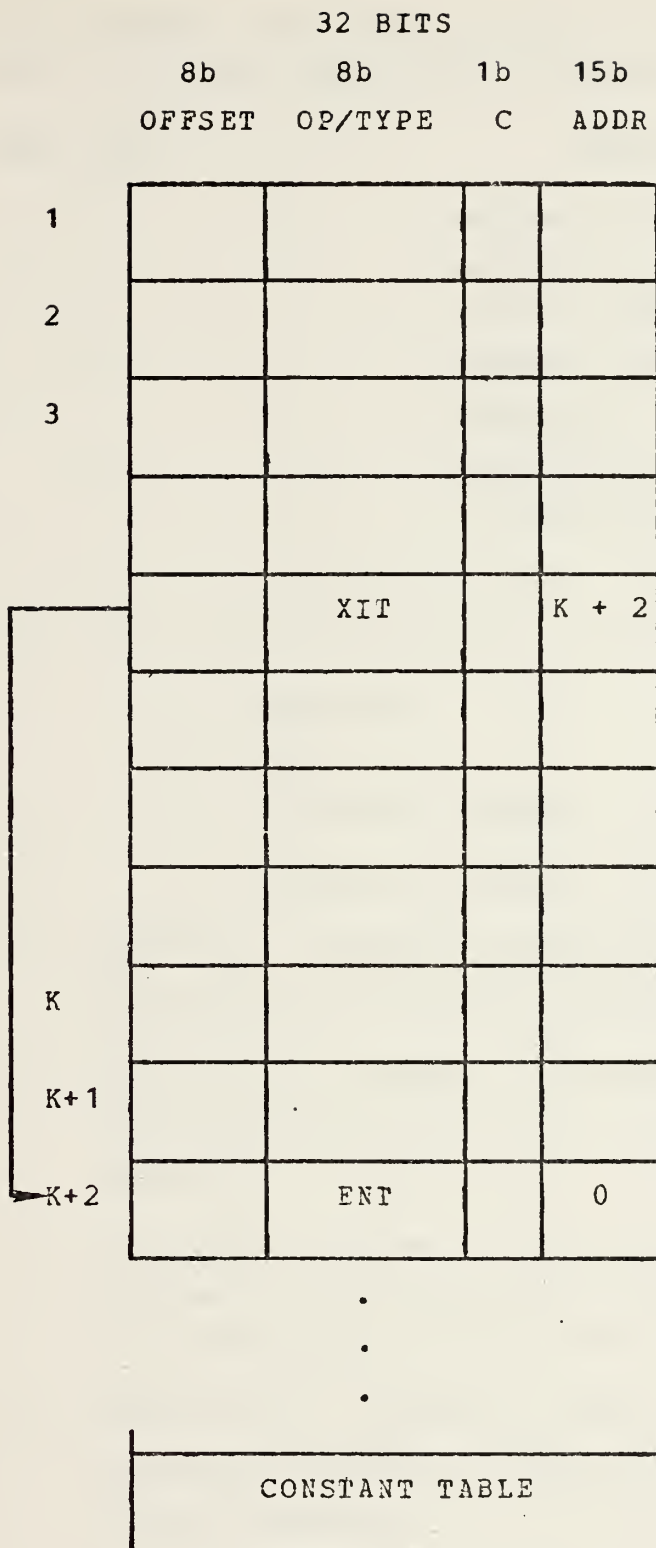


FIGURE 4  
INTERMEDIATE LANGUAGE OUTPUT FORMAT FROM CSFE



The CONSTANT TABLE begins with a count of the number of constants. An entry then follows for each constant as:

<u>BYTE (8b)</u>	<u>CONTENT</u>
0	number of characters for the constant
1	type of constant
2-3	address assigned to the constant
4+i	character string of length i representing the constant.

The basic CSFE procedures are as listed with calling values of:

o - operator  
t - type  
s - character string  
x - integer constant  $0 \leq x \leq 2^{14}$ .

- 1) EMITO(o) - places a reference to an operator in the Polish sequence.
- 2) EMITC(s,t) - places a reference to the constant string s of type t into the Polish sequence.
- 3) EMITA(x) - places a reference to address x into the Polish sequence. Type LOC is associated with the address.
- 4) EMITAC(s,t) - places a reference to the constant string s of type t into the Polish sequence.



- 5) EMITI(t) - places a reference to an operand of type t with an indeterminate value into the Polish sequence.

Additionally, the CSFE handles "branch point resolution." Branch points are referenced symbolically in the Polish sequence during compilation. A particular point in the sequence can be given a symbolic name for future reference. Whenever a symbolic location is defined, an ENT is inserted. Furthermore, references can be made to locations which have not as yet been compiled, called an "unresolved reference." When an unresolved reference occurs, the CSFE "resolves" the reference by inserting an address in the ADDR field of all XIT operands that previously referenced the symbol.

The CSFE procedures for label processing are:

- 1) EMITE(s) - ENT type variable which resolves the top-most stacked reference to s.
- 2) EMITB(s) - backward reference to a label s which has already occurred.
- 3) EMITF(s) - forward branch point which has not occurred.
- 4) EMITPE(s) - ENT type variable that does not resolve any references to label s, called a "push down entry."
- 5) EMITPF(s) - forward reference to label s which "pushes" all previous references to label s. An EMITE(s) will resolve this reference but not the references which were "pushed down."



- 6) SAVEDLAB - saves the current state of the label stack. Used to process inner loops such as embedded for-loops.
- 7) RESTORELAB - restores the label stack to the configuration present upon the last call to SAVEDLAB.

Additional utility procedures are also present as follows:

- 1) EMITR(x) - places a REFER operator into the Polish sequence. The ADDR field is set to x. This is normally called at each card boundary in the source program with a line number of x; therefore, if an error condition is detected, the Polish sequence is scanned backwards to the last REFER. The ADDR field along with the OFFSET field at the point of error are combined to pinpoint the error in the original source program.
- 2) EMITP(x) - places a PASS operator with an ADDR field of x into the Polish sequence. The PASS operator is ignored by the CSF. Thus, the PASS operator can be used to pass information to subsequent modules.
- 3) EMITT(x) - places a TOG operator with an ADDR field of x into the Polish sequence. The x value is a bit word representation of toggle flags that can be set on the compile pass for later detection by CSF. Compilation toggles are enabled or disabled in the ALGOL-E implementation by:





- a) \$CODE - results in the Polish sequence being interlisted with the source program.
- b) \$TABLES - causes CSF to give a full trace by table dumps of the optimization process.
- c) \$EXEC - causes a partial trace of the CSF analysis.
- d) \$TRACE - causes CSF to give a partial trace of the optimization process.

During the interlisting of the Polish sequence, (i.e., \$CODE is selected), each line of the code trace will hold

- 1) OFFSET field - not used
- 2) OPCODE/TYPE - operator or operand type
- 3) C field - constant indicator
- 4) ADDR field - address
- 5) 32-bit word in hexadecimal format.

A "+" before a line indicates the resolution of a forward reference while a "++" indicates that the entry point at that line number has caused the resolution of a forward reference or has been referenced again as a backward reference. The ADDR field of each ENT variable is incremented on each reference from an XIT variable. Therefore, it contains the number of branches into the location of ENT. TABLE IV is an example of the code interlisting option.



The "constant table" is appended to the end of the Polish sequence. This table provides a character string representation of each constant encountered in the source program. The entire code file (i.e., Polish sequence and constant table) is passed to the CSF at the end of compilation.

## B. OPTIMIZING FUNCTIONS INCORPORATED IN THE CSF

Before giving the details of the CSF implementation, it is necessary to describe the optimizing function and meet operator currently used. Information for three forms of optimization is collected:

- 1) Constant Propagation
- 2) Simplifying Formal Identities
- 3) Common Subexpression Elimination.

### 1. Constant Propagation

The pool associated with each basic block is a partition of the program expressions. Constant propagation requires that a constant indicator be associated with every equivalent class. For example:

```
A := 2;  
B := 3;  
C := A + B;
```

could be rewritten as

```
C := 5;
```

since A and B are both known constants. Therefore, the pool associated with each node is a partition with a constant indicator for each equivalence class. The meet operator combines the constant indicators.



TABLE IV

## ALGOL-E CODE INTERLISTED PROGRAM

CARD	BL	SYL	COMMENT	TEST PROGRAM # 10	CARD NUMBER	BLOCK NUMBER
EXECUTE	CODE	LOCATION	OFFSET	OPCODE/TYPE	C-FIELD	POLISH ELEMENT COUNTER
STABLES	LOC	LOC	LOC	LOC	LOC	LOC
5	0	2	BEGIN LOCAL A,B,C,D,E;	LOC	3	00020003
6	1	2	C := 2;	INT	1	00050001
				STO	0	00280000
				DEL	0	002A0000
7	1	6	D := 3;	LOC	4	00020004
				INT	2	00050002
				STO	0	00280000
				DEL	0	002A0000
8	1	10	READ(A,B);	LOC	1	00020001
				INT	0	00050000
				STD	0	00290000
				LOC	2	00020002
				INT	0	00050000
				STD	0	00290000
9	1	16	WHILE A GTR B DO	ENT	0	00030000
				LOC	1	00020001
				LOC	0	00270000
				LOC	2	00020002
				LOC	0	00270000
				GTR	0	00210000
				XIT	0	00040000
				XIT	26	0004001A
				BSC	0	002F0000
				ENT	1	00030001
10	1	26	B := C + D;	LOC	2	00020002
				LOC	3	00020003
				LOC	0	00270000
				LOC	4	00020004
				LOC	0	00270000
				ADD	0	00100000
				STO	0	00280000
				DEL	0	002A0000
				XIT	17	00040011
				ENT	1	00030001
				BRS	0	002E0000
				ENT	0	00030000
				XIT	37	00040025
				ENT	1	00030001
11	1	37	E := D + C;	LOC	5	00020005
				LOC	4	00020004
				LOC	0	00270000
				LOC	3	00020003
				LOC	0	00270000
				ADD	0	00100000
				STO	0	00280000
				DEL	0	002A0000
12	1	45	END			
13	1	45	EOF			

## ALGOL-E SOURCE

```

BEGIN LOCAL A,B,C,D,E;
  C := 2;
  D := 3;
  READ (A,B);
  WHILE A GTR B DO
    B := C + D;
  E := D + C;
END
EOF

```

BRANCH POINT RESOLUTION

CODE FILE COPIED (45 WORDS)  
 CONSTANT TABLE COPIED (4 WORDS)  
 2 RECCROS WRITTEN INTO FILE 1  
 END OF COMPILATION FEBRUARY 9, 1975. CLOCK TIME = 18:2:58.02.

13 CARDS WERE READ.  
 NO ERRORS WERE DETECTED.



## 2. Simplifying of Formal Identities

A list of the simplifying formal identities specified in the CSFG are available in tabular form in the CSF. During the analysis, the expressions are compared with this table and transformations are made if a match is found. For example, the expression

$$A := A + 0;$$

matches the form of the additive identity and could, therefore, be eliminated. Thus, no optimizing function or meet operator need be specified since these transformations are all local.

## 3. Common Subexpression Elimination

As described in Section I. C., the pools for common subexpression elimination consist of partitions of program expressions into groups that are known to have identical values. The meet operator is defined as the intersection on the equivalent classes.

The combination of the above three optimization forms are incorporated into the current CSF. This combination logically allows detection of a fourth optimization form, common subexpressions that are linked to a previous constant propagation. The meet operator is the intersection operator. The four types are detected using a hierarchy of:

- 1) Propagated Constants
- 2) Simplifying Formal Identities
- 3) Common Subexpressions that link to a Propagated Constant
- 4) Common Subexpressions.





## C. CODE SYNTHESIS FILTER (CSF)

The following sections give the programming structures and methodology that are used in the implementation of the global flow analysis algorithm. The CSF module was written in the XPL language utilizing the XCOM control system [37].

### 1. Program Flow

The actual implementation of the global flow analysis algorithm is incorporated in this module. In order to analyze the input Polish sequence, a "meta-execution" is performed on the intermediate language. This process utilizes the "value number" concept of Cocke [10] which results in a more simplified data structure than would normally be expected. The "value number" will be referred to as a "class number," and an equivalence class will have one such class number assigned to it.

The CSF is separated into three logical parts:

- 1) Initialization
- 2) Control Flow Analyzer (CFA)
- 3) Basic Block Analyzer (BBA).

The initialization routines accept the intermediate code file from the CSFE and prepare all data structures for processing. It is here that a CONSTANT TABLE is built as produced by the ALGOL-E compiler.

The CFA controls the flow of processing from block to block, performs the meet operation and terminates the CSF.



The EBA is invoked to analyze each block (i.e., the application of the optimization function). It is here that meta-execution takes place and optimization information is detected. Figure 5 depicts the overall flow of the CSF.

The enclosed listing of CSF has been fully annotated in order to provide documentation of individual routines. A complete discussion of table structures and the more important procedures are given in the following sections.

## 2. Implementation Methodology

The implementation of the global flow analysis algorithm was done using structured programming concepts. All functions are isolated within procedures and each procedure is as autonomous as possible. A complete description of the program organization and output can be accomplished by examining four areas:

- 1) Control Structures
- 2) Meta-Execution
- 3) Meet Operation
- 4) Output Formats.

### a. Control Structures

The program branching structure is followed by a procedure called "CONTROL-FLOW," the Control Flow Analyzer (CFA). The CFA repeatedly cycles through the following steps:

- 1) select a block to be processed  
from the top of the list L.
- 2) call the Basic Block Analyzer (BBA)  
to process the block.



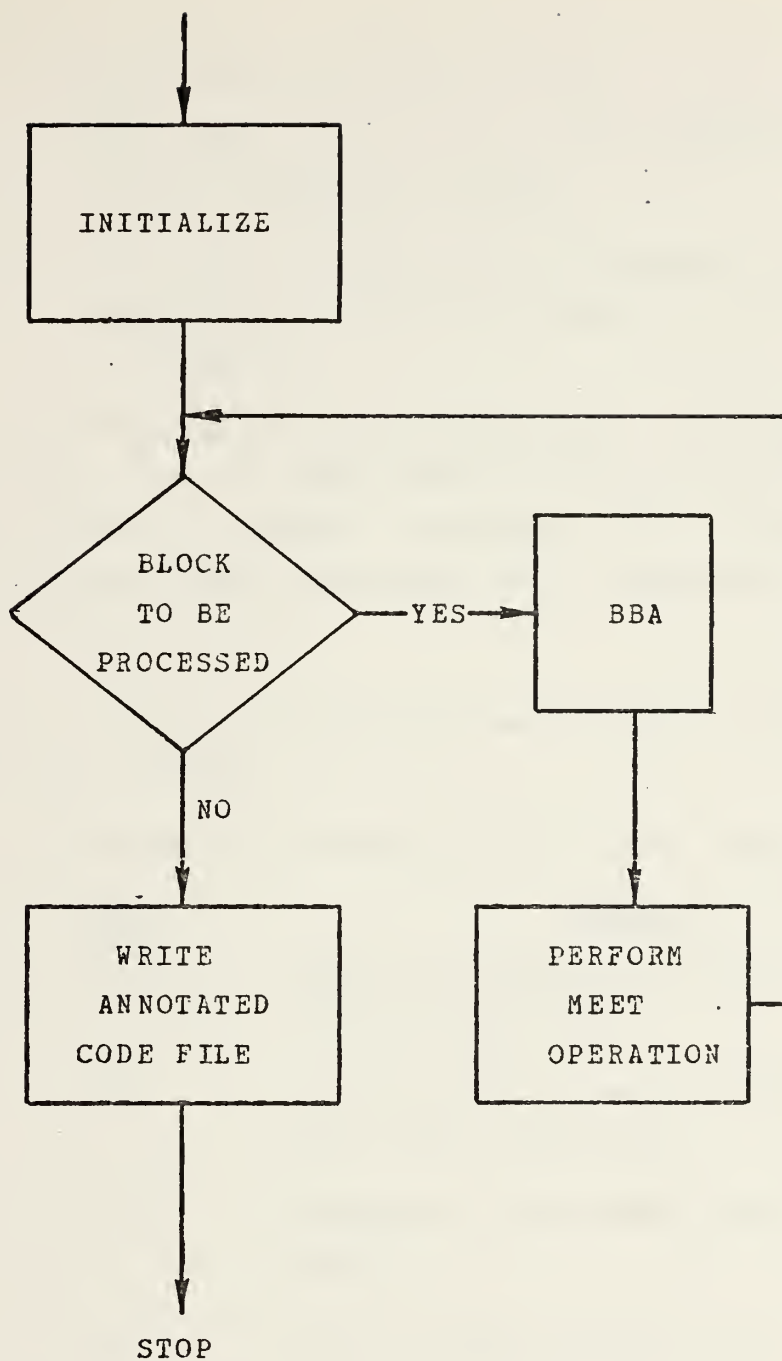


FIGURE 5  
OVERALL FLOW OF CSF



- 3) use the  $P_o$  produced by the  
the BBA to perform the meet operation  
on all successor blocks.
- 4) order the blocks to be processed list  
according to the block selection rule  
in effect.

These steps are continued until no blocks are available to be processed in step 1. The CFA then calls a procedure, "OUTPUT\_OPTIMIZATION," which writes an annotated intermediate language code file along with a synopsis of the optimization process.

The "blocks to be processed list," L, consists of two elements as follows:

- 1) CONTROL is a thirty-two (32) bit entry of:

<u>BITS</u>	<u>CONTENT</u>
0-15	first word address of the block in the code file,
16	set if initial pool has been built for this block,
17-30	immediate predecessor block number,
31	set if block is not to be processed (i.e., $P_c \leq P_i$ in step Q5 of the flow algorithm).





. 2) SAV-TOG is an eight (8) bit entry of:

BIT

- |     |   |
|-----|---|
| 0   | set if control flow trace of optimization process desired,                        |
| 1   | set if complete table trace of optimization process desired,                      |
| 2   | set if detected constant operands are to be propagated,                           |
| 3   | set if false branches are to be removed from detected constant decision branches, |
| 4   | set if output from CSF is desired in punched cards,                               |
| 5-7 | are not currently used.   |

The function of the two tables is to save blocks to be processed along with optimizer toggles for each block. If it is found that a block need not be processed (i.e., the test of  $P_c \leq P_i$  is true after performing the meet operation in step Q5 of the algorithm) a single bit can be set rather than rearranging the table sequence. If an immediate successor block has never been encountered before, its  $P_c$  becomes the  $P_o$  from the immediate predecessor block in the procedure "BUILD\_INPUT\_POOL."

Blocks are entered on the top of the "to be processed list," L as they are encountered by the BBA (i.e., when a branch operator is encountered). After performing the meet operation on all successor blocks, the CFA calls



the procedure "ORDER\_BLOCK\_LIST," which arranges the "to be processed list," L, in a top down sequence for actual processing according to the specified block selection method. Presently, any one of four (4) block selection methods can be specified. The are:

1. Last-in-First-out (LIFO).
2. First-in-First-out (FIFO).
3. Select the  $P_i$  with the minimum number of equivalence classes called the min pool selection method, as discussed by Lukasczyk [34].
4. Select the  $P_i$  with the maximum number of equivalence classes called the max pool selection method. This was included for comparison against the min pool selection method.

In order to control block processing, a "block header" for each block is generated when it is first encountered. This header is updated each time the block is traversed; consequently, it provides a summary for the block at any point during the optimization process. These block headers are saved sequentially above the intermediate language code file. The block header format is:

<u>WORD</u>	<u>CONTENT</u>
0	Block # (15 bits)   previously processed indicator (1 bit)   STATES TABLE index (16 bits)
1	First word address in code file (16 bits)   last word address in code file (16 bits)



- 2                   Number of transversals
- 3                   Number of references
- 4                   Number of predecessor blocks
- .                  referencing this block
- .                  (variable number of
- .                  entries).

Prior to passing a block's first word address to the BBA, the CFA places its  $P_i$  on the ADDRESS and EXPRESSION TABLES using information from the STATES TABLE. These tables are among the primary mechanisms for the meta-execution performed by the BBA and will be discussed in the next section. In this manner, the meta-execution tables are preset for any block thus allowing random processing of blocks.

The meet operation is performed by the CFA through a call to the procedure PERFORM\_MEET\_OP which applies the meet operator to a single immediate successor block of the block just processed. A true response from the PERFORM\_MEET\_OP procedure indicates that the immediate successor block is to be processed (i.e., the test of  $P_c \leq P_i$  in step Q5 of Algorithm Q). As was mentioned previously, a single bit in the CONTROL TABLE entry for each successor block is then set or cleared to indicate future processing. The meet operation will be discussed in more detail in Section II.C.2.c.

#### b. Meta-Execution

The meta-execution is performed within the procedure BASIC\_BLOCK which is the control routine for the BBA. The Polish sequence is symbolically evaluated using an



EXECUTION STACK. This process uses "class numbers" instead of the actual value of a variable since the actual value is not generally known at compile time. The EXECUTION STACK has four (4) elements:

- 1) EXVAL# - a sixteen bit entry which holds the class number.
- 2) EXTYPE - an eight bit entry which holds the type of an operand (e.g., BOOL or INT).
- 3) EXCON - a one bit indicator to identify an entry as a constant.
- 4) EXADD - a sixteen bit entry that points to the ADDRESS TABLE entry that corresponds to this EXECUTION STACK entry.

Three other table structures are used in conjunction with the EXECUTION STACK. They are the ADDRESS TABLE, the EXPRESSION TABLE and the CONSTANT TABLE.

The ADDRESS TABLE defines all variables. There are five (5) entries per variable:

- 1) ADDRESS - a thirty-two bit entry that holds an internal code that identifies each variable. This code is generated in sequence by the ALGOL-E compiler beginning with the first declared. This field will be zero if an ADDRESS TABLE entry defines an expression.
- 2) ADDTYPE - an eight bit entry that defines the type of the variable.





- 3) ADDVAL - a sixteen bit entry that holds the current class number assigned to the variable.
- 4) ADDCON - a sixteen bit entry that addresses the proper entry in the CONSTANT TABLE, if the current value of a variable is a constant.
- 5) ADDNAM - a sixteen bit index to the EXPRESSION TABLE if the entry defines an expression. In this case, the ADDRESS entry would be equal to zero.

The CONSTANT TABLE structure consists of four (4) elements that define each constant. The elements are:

- 1) CONSYM - a character string representation of the constant.
- 2) CONTYPE - an eight bit entry that defines the type for the constant.
- 3) CONVAL a sixteen bit entry that holds the class number assigned to the constant.
- 4) CONINT - a thirty-two bit binary representation of the actual constant value.

The EXPRESSION TABLE (VTAB) is structured as a linear table with sequential expression entries made as they occur. A hash coded retrieval method is used to access the table resulting in rapid search for redundant expressions. The lower portion of VTAB up to HASHBASE is saved for the primary hash link. Access to this link is formed by



INDICATOR + OPERATOR + OPERAND<sub>0</sub> class number +

OPERAND<sub>1</sub> class number +...+ OPERAND<sub>n</sub> class number.

The link address obtained will either hold a zero indicating no expression with this hashcode has been encountered, or an index to the most recent expression that hashed to this address. The INDICATOR in the hash formula is used to determine where the class numbers for the operands can be found. Specifically, the expression is located either in the EXECUTION STACK or in the EXPRESSION TABLE. An EXPRESSION TABLE entry is structured as:

<u>WORD</u>	<u>CONTENT</u>
0	Bits 16-31 hold the original Hash Link address. Bits 0-15 hold the operator code.
1	Hash code collision field which is zero (0) if no further expression links are present.
2	Class number for operand <sub>0</sub> .
.	Class number for operand <sub>1</sub>
.	(variable number of operands
.	allowed).
2 + I	Class number for resultant operand <sub>0</sub> .
2 + I + 1	Class number for resultant operand <sub>1</sub> (variable number of
.	resultant operands allowed).
.	
2 + I + J + 1	Master chain pointer links all entries. Used for bookkeeping.



Additionally, the upper portions of the ADDRESS TABLE and EXPRESSION TABLE are used to hold the  $P_c$  for a block, as will be further explained in Section II.C.2.c.

The meta-execution operates by loading Polish operands onto the EXECUTION STACK and setting indexes to their respective ADDRESS TABLE entries. When an operator is detected in the Polish sequence, the EXPRESSION TABLE is searched for an identical entry using the hashcode chain. In this manner, common subexpressions are easily detected.

Because of the link from the ADDRESS TABLE to the CONSTANT TABLE, the detection of constant propagation is a simple matter of bit checking on the operands of an expression. If all operands are in fact constants, a propagated constant has been detected.

The application of simplifying formal identities is done by a comparison against a table of simplification rules generated by CSFG for each expression. Both propagated constants and simplification detection are accomplished by a call to the procedure SIMPLIFY by the BBA. The generated constant propagation case statement shown in TABLE III, is located in the procedure PROPAGATE which is invoked when a propagated constant is detected by the procedure SIMPLIFY. PROPAGATE then computes the propagated constant as specified by the operator. The constant value is then entered into the CONSTANT TABLE if it does not already exist.



As possible code transformations are detected, they are also flagged in the table OPTIM\_TYPE which has a parallel entry for each code word. The entries in the OPTIM\_TYPE TABLE are coded as:

<u>CODE</u>	<u>MEANING</u>
0	no transformation present
1	simplification possible
2	propagated constant
3	common subexpression
4	common subexpression that links to a propagated constant.

Additionally, an optimization entry is saved for future reference at the extreme top of the EXPRESSION TABLE (VTAB). The address of this optimization entry is entered into the ADDR field of the intermediate code for use by later passes. The form of this entry is:

<u>VTAB LOCATION</u>	<u>CONTENTS</u>
VTOPR	operand <sub>0</sub> class number
.	(variable number of
.	operands),
.	
VTOPR - X	code location at which an identical expression can be found. Zero (0) if not applicable,
VTOPR - X - 1	resultant operand class number,

where VTOPR is the current pointer to the last entry in the top of VTAB. Upon subsequent passes for a block, this





optimization expression entry will be updated as necessary. If a previous optimization does not recur on a subsequent pass, the indicator in OPTIM\_TYPE is reset to zero (0), but the optimization expression entry remains linked.

During the BBA, the tables generated by the CSFG come into use. The linkage between tables is in some cases rather complex and needs clarification for a complete understanding of the CSF.

In order to identify the operator codes in the code file, two (2) tables are used:

- 1) OPRANGE
- 2) OPCODES.

OPRANGE is a table that is indexed by the OPCODE field in the code file. Each entry addresses the first character for a particular OPCODE in the character string called OP\_CODES. Figure 6 illustrates this relationship.

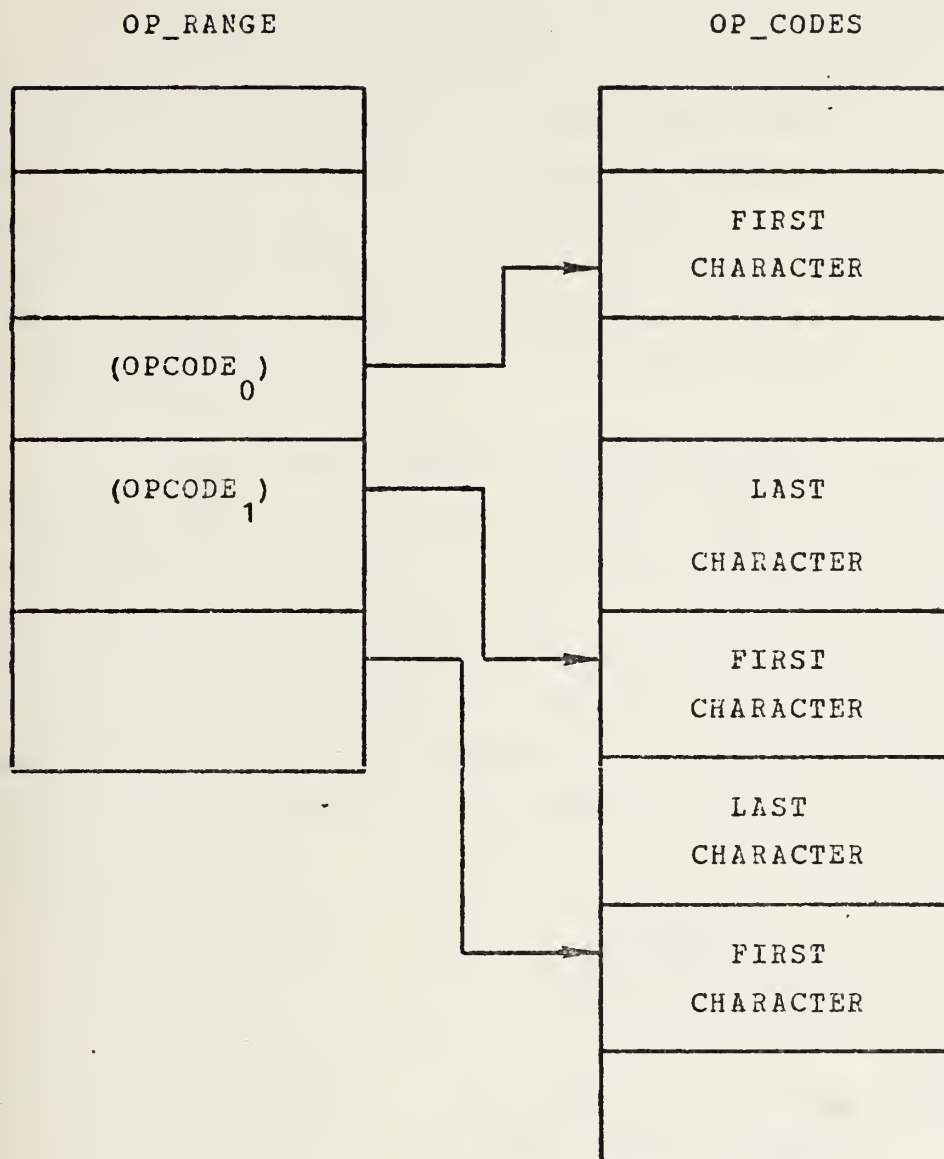
To assist in the processing of expressions, five (5) tables are used:

- 1) OP\_DEGL
- 2) OP\_DEGR
- 3) OP\_INDEX
- 4) OP\_INFO
- 5) OP\_STR.

OP\_DEGL is indexed by OPCODE and contains the number of operands necessary for an operator.

OP\_DEGR is indexed by OPCODE and contains the number of result operands that the operator will generate.





**FIGURE 6**  
**OP\_RANGE TO OP\_CODES RELATIONSHIP**



OP\_INDEX is indexed by OPCODE and points to the first entry in OP\_INFO for an operand. OP\_INFO holds the type code for each operand and resultant operand up to entry ninety-five (95) in the ALGOL-E implementation. As noted in Figure 7, more than one configuration of operand and resultant operand types can exist for a particular operator. The OP\_DEGL and OP\_DEGR counts provide a means of breaking a multiple configuration into its exclusive parts.

For the ALGOL-E implementation, the OP\_INFO TABLE above entry ninety-five (95) is used in conjunction with the SIMP\_INDEX TABLE and the OP\_STR TABLE to identify simplification rules. SIMP\_INDEX is indexed by the operator code and indexes the first simplification rule in OP\_INFO for that operator. Table OP\_STR holds the string representation of the identity operands "1" and "0" for ALGOL-E. Each entry of OP\_INFO is eight (8) bits as follows:

<u>BIT</u>	<u>CONTENT</u>
0	1 = identity operand 0 = variable identifier
1-7	if identity operand this field points to OP_STR for proper identity. If variable operand this field holds the type code.



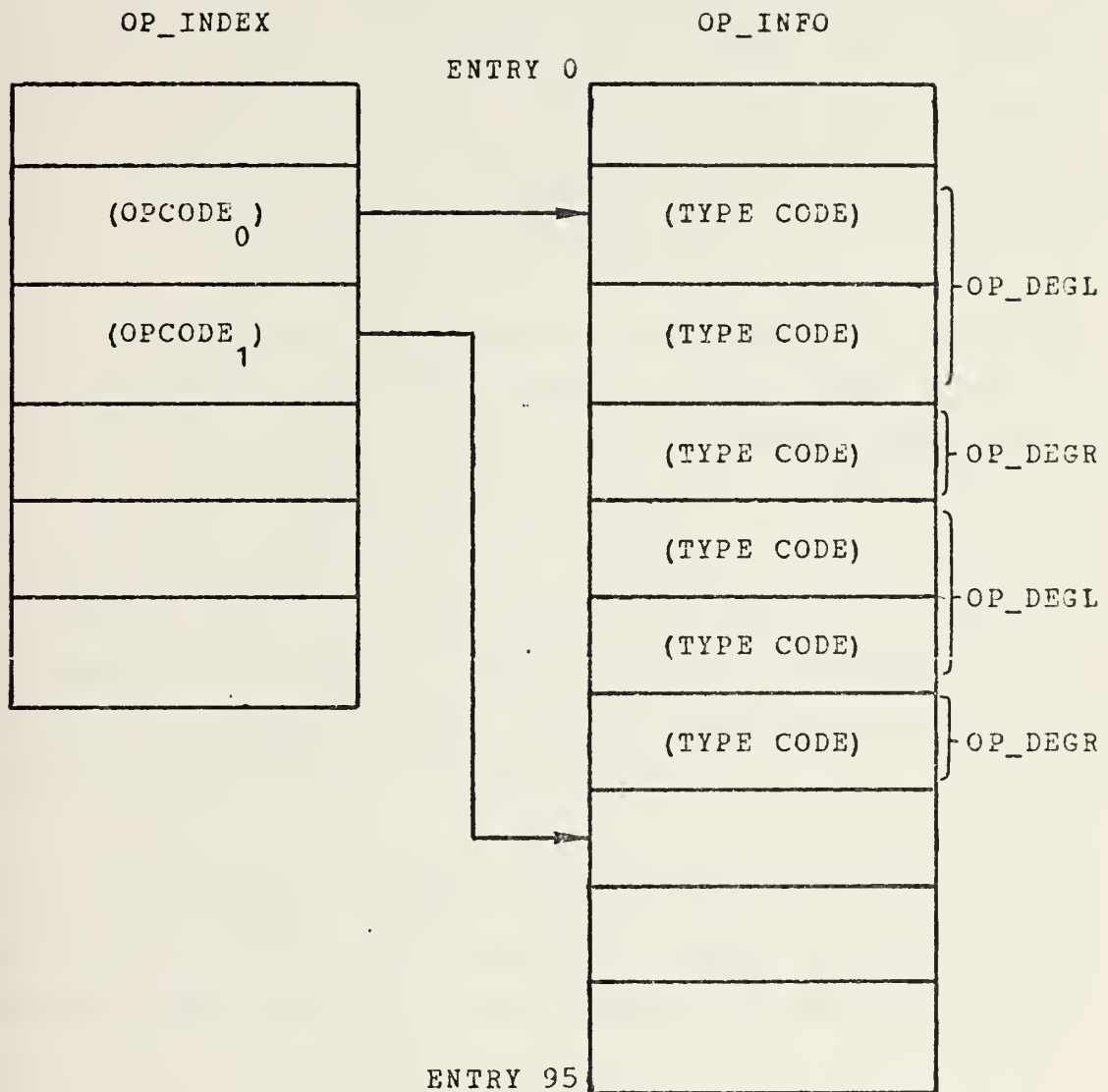


FIGURE 7  
OP\_INDEX TO OP\_INFO RELATIONSHIP





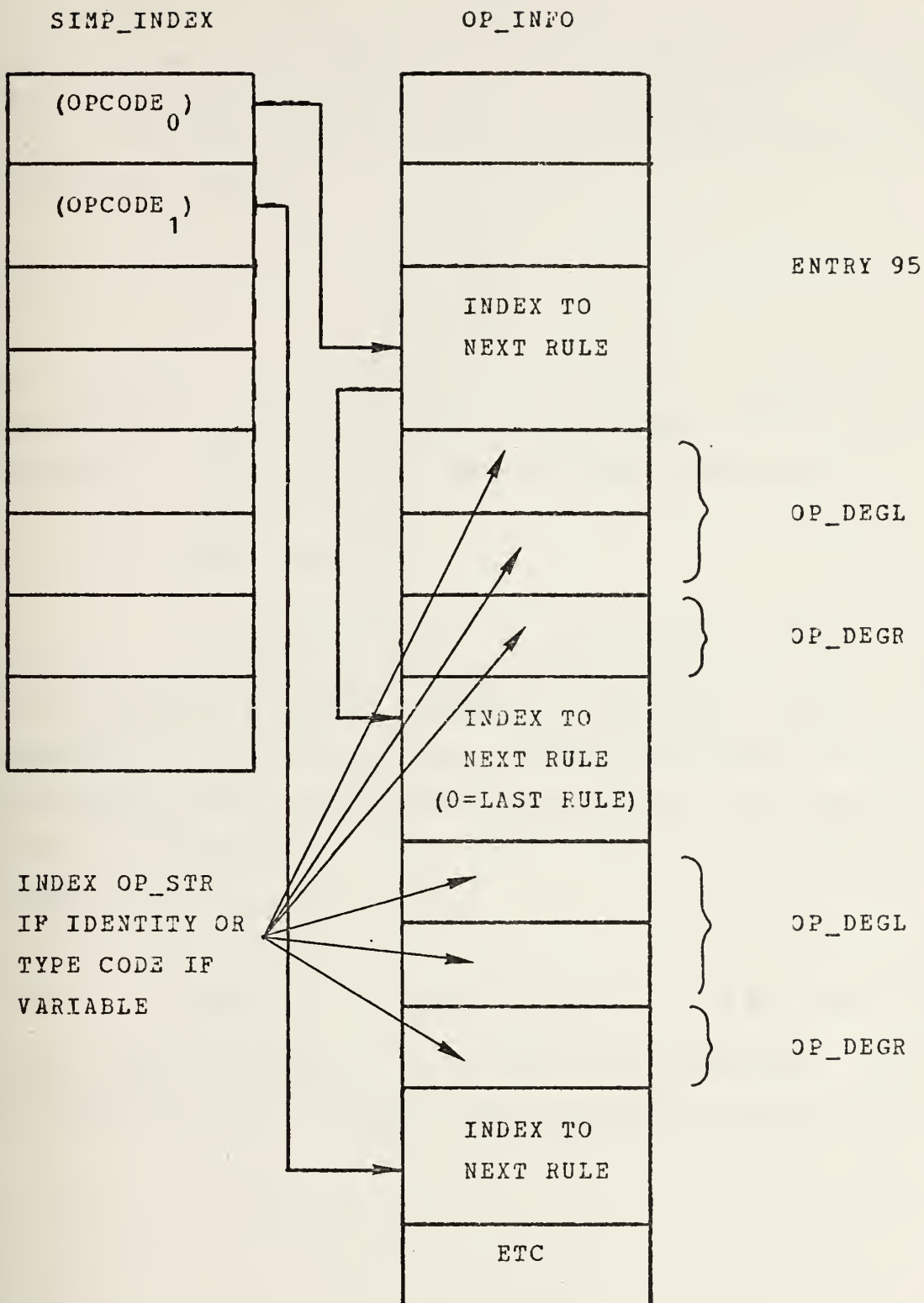
Each entry of OP\_INFO is a description of an operand or a resultant operand. Figure 8 depicts the Simplification Link Structure within OP\_INFO. Note that more than one rule can exist for an operator in OP\_INFO. A link from rule to rule is present, and OP\_DEGR and OP\_DEGL are used to process each rule. The last link field contains zero (0) which serves as the rule terminator for an operator. When a rule is matched against an existing expression, a simplification has been found.

The BBA sequentially processes the Polish elements beginning with the block's first word address provided by the CFA. Since the BBA processes only basic blocks, it returns control to the CFA upon encountering a branch operator or an ENT type variable. Any XIT operands encountered by the BBA result in entries on the CONTROL STACK with the ADDR field specifying the first word address of the immediate successor block. An ENT variable also results in a CONTROL STACK entry beginning at the next sequential instruction in the code file.

Because of constant propagation, the BBA is able to alter the actual block flow of the program if a propagated constant is detected while processing a conditional branch operator. For example, given the program

```
A := 5;  
B := 10;  
IF A > B THEN  
C := 6;  
D := 7;
```





**FIGURE 8**  
SIMPLIFICATION LINK STRUCTURE



the test of  $A > B$  would obviously always be false. When this condition is detected, only one of the XIT variables generated by the compiler would be placed on the CONTROL STACK. This can result in entire sections of a program never being traversed. The flow of the above example would effectively become

```
A := 5;
B := 10;
D := 7;
```

This facility can be controlled by a toggle called BRANCHTOG. If it is false, both exits will always be processed. Currently it is set to a true condition.

#### c. Meet Operation

The meet operation is accomplished from within procedure PERFORM\_MEET\_OP which is called by the CFA. Each call results in a single immediate successor block being processed. A true or false response indicates to the CFA whether the immediate successor block is to be marked for future processing. That is, if  $P_c \leq P_i$ , then PERFORM\_MEET\_OP returns false.

The  $P_c$  for a block is saved via the STATES TABLE in the upper portions of the ADDRESS and EXPRESSION TABLES. The STATES TABLE is a linear table which is structured as:



<u>WORD</u>	<u>CONTENT</u>
0	0 always zero
1	0
2	number of states for block #1
.	(index to ADDRESS table
.	for each state)
.	.
2 + I + 1	0
2 + I + 2	0
2 + I + 3	number of states for block #2
etc	etc

where a "state" is a single variable or expression description. The block header for each block points to its respective STATES TABLE entry. The ADDRESS TABLE entry for the  $P_c$  is nearly identical to the format previously discussed; however, it is saved linearly from the top down instead of from the bottom up. The EXPRESSION TABLE entry is also slightly different. An ADDRESS TABLE entry corresponding to an expression entry will still have zero (0) in the ADDRESS entry. However, the ADDNAM pointer will address the top of VTAB instead of the bottom. In VTAB, only the operands for the expression will be found. Note, finally, that the complete state of any one block is distinct from the other blocks; that is, no ADDRESS TABLE state entry is shared by blocks.

An important point in understanding the structure of the EXPRESSION TABLE (VTAB) is that it holds three separate entries. They are:





- 1) current expressions for the block being processed,
- 2) information on detected transformations, and
- 3) operands for the saved current optimized pools of all blocks.

Only the current expression entries are hash coded, and reside in the lower part of VTAB. The other two entry types are found in the upper regions of VTAB.

Prior to calling PERFORM\_MEET\_OP, the CFA structures the lower ADDRESS TABLE so that it completely reflects the  $P_o$  of the last processed block. This is accomplished by moving any EXECUTION STACK entries onto the ADDRESS TABLE. The ADDRESS field will be a number representing the relative position of the entry on the EXECUTION STACK. The uppermost bit (bit 15) of the ADDRESS field is set to identify the entry as originating from the EXECUTION STACK. Therefore, the meet operation need only look at the lower ADDRESS TABLE for the  $P_o$  and the upper ADDRESS TABLE specified by the STATES TABLE for the  $P_c$  of the immediate successor block.

The meet operation is accomplished in three (3) steps:

- 1) Initialize
- 2) Process all simple variables
- 3) Process all expressions.

Two (2) tables, the CLASS OCCURRENCE LIST and the CLASS REFINEMENT TABLE, are the primary structures used to accomplish the meet operation as developed by Kildall [28].



The initialization stage builds the CLASS OCCURRENCE LIST based on the  $P_c$  of the block to be processed. This list is simply a count of the occurrence of each class number that is assigned to a simple variable or an expression.

After the CLASS OCCURRENCE LIST is built, all simple variables are processed from the  $P_c$  of the immediate successor block. The  $P_o$  is searched for each identifier in the  $P_c$ . If the identifier exists, the class numbers from the two (2) pools are entered into the CLASS REFINEMENT LIST. If the class numbers are not the same, a new "refined class number" is assigned to the identifier. The identifier is then placed in the  $P_i$  with the refined class number. As each identifier is processed from the  $P_c$ , its class occurrence is decremented in the CLASS OCCURRENCE LIST.

After all simple variables have been processed, each expression in the  $P_c$  is considered. When an unprocessed expression is found, a check is made to ensure that the the CLASS OCCURRENCE LIST for all its operand class numbers are zero (0). If they are not, the expression is not processed until they are, with a single exception. Simplification rules such as

$$A + 0 = A$$

cause both  $(A + 0)$  and  $A$  to appear in the same class. When this occurs,  $(A + 0)$  is treated the same as  $A$ .

If an expression can be processed, its resultant class number count is decremented in the CLASS OCCURRENCE



LIST. A list of candidate expressions to consider in the  $P_o$  is formed by examining the CLASS REFINEMENT LIST. If an expression in the  $P_c$  has operands  $OP_1$  and  $OP_2$ , a search of the CLASS REFINEMENT LIST will be made to identify class numbers in the  $P_o$  that are equivalent to  $OP_1$  and  $OP_2$ . For example, if an output pool is

$$P_o = \{A \mid B \mid A + B, B + A\}$$

(1) (2) (3)

and a current pool is

$$P_c = \{A \mid B \mid A + B, B + A\}$$

(5) (6) (4)

where each class is labeled with a class number. The CLASS OCCURRENCE LIST would be:

<u>NUMBER OF OCCURRENCES</u>	<u>CLASS NUMBER</u>
1	4
1	5
1	6

After processing simple variables, the CLASS OCCURRENCE LIST would be:

<u>NUMBER OF OCCURRENCES</u>	<u>CLASS NUMBER</u>
1	4

and the CLASS REFINEMENT TABLE would be

$P_o$	<u>CLASS NUMBER</u>	$P_c$	<u>CLASS NUMBER</u>	<u>REFINED CLASS NUMBER</u>
	1		5	7
	2		6	8



Since identifiers A and B were both found, the  $P_i$  to this point would be

$$P_i = \{ A \mid B \}.$$

(7) (8)

Upon searching the CLASS OCCURRENCE LIST for the operand class numbers of the expression (A + B) in the  $P_c$ , none will be found; therefore, the expression can be processed. The resultant class number, 4, is deleted from the CLASS OCCURRENCE LIST. Using the CLASS REFINEMENT TABLE a list of candidate expressions to search for in the  $P_o$  is constructed. This list is developed by taking the operands of the expression in the  $P_c$  and searching the CLASS REFINEMENT TABLE for matches in the  $P_o$ . For example, a class number of 1 in the  $P_o$  is equivalent to a class number of 5 in the  $P_c$  and a class number of 2 in the  $P_o$  is equivalent to a class number of 6 in the  $P_c$ . Therefore, the expressions to be searched for would be

$P_o$  EXPRESSION LIST

(5) + (6)

(6) + (5)

since the operator "+" is commutative. The  $P_o$  is then searched for an occurrence of these expressions. If found, the resulting class numbers of the  $P_c$  and  $P_o$  expressions are entered in the CLASS REFINEMENT TABLE in the same manner as for an identifier.





<u>P<sub>o</sub></u> <u>CLASS NUMBER</u>	<u>P<sub>c</sub></u> <u>CLASS NUMBER</u>	<u>REFINED CLASS NUMBER</u>
1	5	7
2	6	8
3	4	9

An expression entry can then be made in the  $P_i$  using the refined class numbers for both operands and resultant operands. The  $P_i$  would then be

$$P_i = \{ A \mid B \mid A + B, B + A \}.$$

(7) (8) (9)

After all expressions have been processed, the CLASS OCCURRENCE LIST will be empty. If an expression or identifier in the  $P_c$  is not found in the  $P_o$ , the current state of  $P_c$  has been altered and PERFORM\_MEET\_CP will return true, indicating the block must be traversed again.

A problem with the handling of constants occurs if an identifier or expression is a constant in the  $P_c$  and is present in the  $P_o$  but its value is not a constant in the  $P_o$ . The state has changed in this situation, thus requiring a true response from PERFORM\_MEET\_OP.

As each identifier and expression is processed, the  $P_c$  area at the top of the ADDRESS TABLE is updated. This is the  $P_i$  that is being formed, which is to be loaded onto the lower ADDRESS TABLE by the CFA before calling the BBA to process the block. At the completion of the CSF, the



$P_C$  held within the upper ADDRESS TABLE is the  $P_F$  (i.e., MOP) for each block.

#### d. Output Formats

A number of table formats have already been discussed; however, further output format definitions for the equivalence classes and the final optimization information is necessary.

The state elements have one of three formats:

- 1) A (X, Y, Z)
- 2) E (X, Y, Z)
- 3) C (X, Y, Z).

Form A(X, Y, Z) refers to a simple identifier and

X is the identifier code number,

Y is the class number of the constant  
if identifier X is equal to a constant,  
and

Z is the class number assigned the identifier  
X in the ADDRESS TABLE.

Form E(X, Y, Z) refers to an EXECUTION STACK entry and

X is the relative location of the entry from  
the top of the EXECUTION STACK. Y and  
Z are the same as for a simple identifier.

Form C(X, Y, Z) refers to an expression entry and

X represents the class numbers of all operands  
separated by colons,

Y is the class number of the constant



if the resultant class number is  
equal to a constant, and

Z is the class number of the resultant operand.

Another format for expressions is used when the  
expression stack is output at the end of processing each  
block. Its form is:

C (X, Y, Z)

where

X is the string representation of the  
expression operator,

Y is the same as for X in state expressions, and

Z is the same as for Z in state expressions.

The final optimization results are output in the  
form

Optimization at X  
(a<sub>1</sub>, a<sub>2</sub>, ..., a<sub>k</sub> | b, c)

where

X is the location in the intermediate  
code file where the optimization  
was detected,

a<sub>1</sub>, a<sub>2</sub>, ..., a<sub>k</sub> are the class  
numbers of all operands,

b is the address of a previous expression  
that is identical, if one exists, and

c is the resultant class number of the expression.

In all of the above formats, an unknown element will be



printed as "\*."

The block headers are output at the end of the optimization process. The format for each block is:

<u>LINE</u>	<u>CONTENT</u>
1	block number
2	first and last word addresses within the intermediate code file for the block
3	count for the number of times the block was traversed
4	block number of an immediate predecessor block
4 + I	(variable number of predecessor blocks possible).

The intermediate code file is also printed, and the C field indicates the actual location of any optimization information. The OPTIM\_TYPE TABLE is output concurrent with the code file to indicate the optimization type.

Several examples of CSF output are enclosed in which different block selection methods were used as indicated on the output. TEST program number six (6) has all four block selection methods listed so that a comparison of the CSF flow using different block selection methods can be seen.

TABLE V is the CSF output from the source program shown in TABLE IV with \$TABLE selected. The different tables are identified as they occur.





## TABLE V

## TABLE TRACE OUTPUT FROM CSF

\*\*\*CODE SYNTHESIS FILTER\*\*\*

BLOCK SELECTION METHOD WILL BE STEEPEST DESCENT (MINIMUM CURRENT POOL)

\*\*\*COMPLETE TABLE DUMP\*\*\*

DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

\*\*\*TABLE DUMP\*\*\* AT LOCATION = 0

CONSTANTS	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
CONSYM	2	3												
CONTYPE	INT	INT												
CONVAL	46	47												
CONINT	2	3												

← CONSTANT TABLE FROM CSFE

ADDR TABL	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
(EMPTY)														

← SIMPLE VARIABLES STACK IS EMPTY

VALUE STACK (TOP AT 127)	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>

← EXPRESSION STACK IS EMPTY

CONT STK	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
BLOCK#	0													
ENTRY	0													

← SET TO PROCESS BLOCK #1  
STARTING AT LOCATION 0

EXEC STAC	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
(EMPTY)														

← EXECUTION STACK IS EMPTY

\*\*\*\* FORMATTED INTERMEDIATE CODE DUMP \*\*\*\*

LOC	OFF	OP CODE	CNS ADR	RAW CODE	OPTIMIZATION
SET			/ETC		TYPE
0	0	ENT	C 46	003802E	
1	0	TOGGLE	257	00080101	
2	0	TOGGLE	258	00080102	
3	0	LOC	3	00020003	
4	0	INT	1	00050001	
5	0	STO	0	00280000	
6	0	DEL	0	002A0000	
7	0	LOC	4	00020004	
8	0	INT	2	00050002	
9	0	STO	0	00280000	
10	0	DEL	0	002A0000	
11	0	LOC	1	00020001	
12	0	INT	0	00050000	
13	C	STD	0	00290000	
14	0	LOC	2	00020002	
15	0	INT	0	00050000	
16	0	STD	0	00290000	
17	0	ENT	1	00030001	
18	0	LOC	1	00020001	
19	0	LOD	0	00270000	
20	0	LOC	2	00020002	
21	0	LOD	0	00270000	
22	0	GTR	0	00210000	
23	0	XIT	37	00040025	
24	0	XIT	26	0004001A	
25	0	BSC	0	002F0000	
26	0	ENT	1	00030001	
27	0	LOC	2	00020002	
28	0	LOC	3	00020003	
29	0	LOD	0	00270000	
30	0	LOC	4	00020004	
31	0	LOD	0	00270000	
32	0	ADD	0	00100000	
33	0	STO	0	00280000	
34	0	DEL	0	002A0000	
35	0	XIT	17	00040011	
36	0	PRS	0	002F0000	
37	0	ENT	1	00030001	
38	0	LOC	5	00020005	
39	0	LOC	4	00020004	
40	0	LOD	0	00270000	
41	0	LOC	3	00020003	
42	0	LOD	0	00270000	
43	0	ADD	0	00100000	
44	0	STO	0	00280000	
45	0	DEL	0	002A0000	

← INITIAL ENT IS FORCED BY THE  
INITIALIZATION ROUTINES



# TABLE V (CON'T)

BASIC BLOCK #1  
BEGINNING AT 0, ENDING AT 0  
BLOCK TRAVERSED 0 TIMES  
0 REFERENCES: ← BLOCK HEADER

\*\*\*END OF COMPLETE TABLE DUMP\*\*\*

B+2 0 |TOGGLE | 1258 | 000B0102 | ← TRACE IN BBA AT LOCATION 2

\*\*\*TABLE DUMP\*\*\* AT LOCATION = 2

INPUT POOL FOR BLOCK# 1 IS:  
(NULL) ← BLOCK #1 HAS A NULL INPUT POOL

ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

VALUE STACK (TOP AT 127)

CONT STK <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

B+3 0 |LOC | 13 | 00020003 |

\*\*\*TABLE DUMP\*\*\* AT LOCATION = 3

ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
ADDRESS 3  
ADDTYPE NULL  
ADDVAL 0  
ADDCON 0  
← LOC OPERATOR HAS RESULTED IN A SIMPLE VARIABLE ENTRY BEING MADE IN ADDRESS TABLE

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
EXCON N  
EXTYPE LOC  
EXADD 3  
EXVAL# 0  
← EXECUTION STACK HOLDS ONE ENTRY OF TYPE LOC WHICH IS REFERENCING SIMPLE VARIABLE 3 (C)

B+4 0 |INT | 11 | 00050001 |

\*\*\*TABLE DUMP\*\*\* AT LOCATION = 4

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
EXCON N  
EXTYPE LOC  
EXADD 3  
EXVAL# 0  
← THE CONSTANT OF 2 IS NOW ON THE TOP OF THE EXECUTION STACK. NOTE THE CONSTANT INDICATOR AND THE CLASS NUMBER.

B+5 0 |STO | 10 | 00280000 |

\*\*\*TABLE DUMP\*\*\* AT LOCATION = 5

ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
ADDRESS 3  
ADDTYPE INT  
ADDVAL 46  
ADDCON 1  
← THE VARIABLE 3 IS NOW SET TO POINT AT THE CONSTANT 2.  
← POINTS TO CONSTANT TABLE

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
EXCON Y  
EXTYPE INT  
EXADD 1  
EXVAL# 46

B+6 0 |DEL | 10 | 002A0000 |

\*\*\*TABLE DUMP\*\*\* AT LOCATION = 6

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)



TABLE V (CON'T)

B+7 0 |LOC | 14 | 00020004 |

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 7  
 ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
 ADDRESS 3 4  
 ADDTYPE INT NULL  
 ADDVAL 46 0  
 ADDCON 1 0

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
 EXCON N  
 EXTYPE LOC  
 EXADD 4  
 EXVAL# 0

B+8 0 |INT | 12 | 00050002 |

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 8  
 EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
 EXCON N  
 EXTYPE LOC INT  
 EXADD 4 2  
 EXVAL# 0 47

B+9 0 |STD | 10 | 00280000 |

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 9  
 ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
 ADDRESS 3 4  
 ADDTYPE INT INT  
 ADDVAL 46 47  
 ADDCON 1 2

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
 EXCON Y  
 EXTYPE INT  
 EXADD 2  
 EXVAL# 47

B+10 0 |DEL | 10 | 002A0000 |

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 10  
 EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
 (EMPTY)

B+11 0 |LOC | 11 | 00020001 |

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 11  
 ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
 ADDRESS 3 4 1  
 ADDTYPE INT INT NULL  
 ADDVAL 46 47 0  
 ADDCON 1 2 0

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
 EXCON N  
 EXTYPE LOC  
 EXADD 1  
 EXVAL# 0

B+12 0 |INT | 10 | 00050000 |

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 12  
 EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
 EXCON N  
 EXTYPE LOC INT  
 EXADD 1 0  
 EXVAL# 0 48

B+13 0 |STD | 10 | 00290000 |





TABLE V (CON'T)

```

***T A B L E   D U M P*** AT LOCATION = 13
ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
ADDRESS    3    4    1
ADDTYPE    INT  INT  INT
ADDOVAL    46   47   48
ADDOCON    1    2    0

```

```

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
(EMPTY)

```

```

B+14  0  |LOC      | 12      | 00020002 |

```

```

***T A B L E   D U M P*** AT LOCATION = 14
ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
ADDRESS    3    4    1    2
ADDTYPE    INT  INT  INT  NULL
ADDOVAL    46   47   48   0
ADDOCON    1    2    0    0

```

```

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
EXCON      N
EXTYPE     LOC
EXADD      2
EXVAL#     0

```

```

B+15  0  |INT      | 10      | 00050000 |

```

```

***T A B L E   D U M P*** AT LOCATION = 15
EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
EXCON      N
EXTYPE     LOC  INT
EXADD      2    0
EXVAL#     0   49

```

```

B+16  0  |STD      | 10      | 00290000 |

```

```

***T A B L E   D U M P*** AT LOCATION = 16
ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
ADDRESS    3    4    1    2
ADDTYPE    INT  INT  INT  INT
ADDOVAL    46   47   48   49
ADDOCON    1    2    0    0

```

```

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
(EMPTY)

```

```

B+17  0  |ENT      | 11      | 00030001 |

```

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 2 IS:  
A(2,\*,49) A(1,\*,48) A(4,47,47) A(3,46,46)

THE INITIAL CURRENT  
POOL FOR BLOCK #2 IS THE  
OUTPUT POOL FROM BLOCK #1

```

***T A B L E   D U M P*** AT LOCATION = 17
CONT STK <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
BLOCK#    1
ENTRY     17

```

BLOCKS TO BE PROCESSED LIST HAS A SINGLE  
ENTRY FROM BLOCK #1 TO LOCATION 17.

```

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
(EMPTY)

```

CONTROL AT 17, FROM BLK 1 TO BLK 2 (PASS 1)  
C+17 0 |ENT | 150 | 00038032 |

CFA PASSES CONTROL FROM BLOCK #1 TO  
BLOCK #2.

```

***T A B L E   D U M P*** AT LOCATION = 17

```

INPUT POOL FOR BLOCK# 2 IS:  
A(2,\*,49) A(1,\*,48) A(4,47,47) A(3,46,46)

INPUT POOL FOR BLOCK #2..STATE IS FOUR SIMPLE  
VARIABLES, TWO OF WHICH ARE CONSTANTS.

```

ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
ADDRESS    3    4    1    2
ADDTYPE    INT  INT  INT  INT
ADDOVAL    46   47   48   49
ADDOCON    1    2    0    0

```





TABLE V (CON'T)

VALUE STACK (TOP AT 127)

EXPRESSION STACK IS EMPTY

CONT STK <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

B+18 0 |LOC | 11 | 00020001 |

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 18  
EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
EXCON N  
EXTYPE LOC  
EXADD 1  
EXVAL# 0

B+19 0 |LOD | 10 | 00270000 |

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 19  
EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
EXCON N  
EXTYPE INT  
EXADD 0  
EXVAL# 48

B+20 0 |LOC | 12 | 00020002 |

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 20  
EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
EXCON N N  
EXTYPE INT LOC  
EXADD 0 2  
EXVAL# 48 0

B+21 0 |LOD | 10 | 00270000 |

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 21  
EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
EXCON N N  
EXTYPE INT INT  
EXADD 0 0  
EXVAL# 48 49

B+22 0 |GTR | 10 | 00210000 |

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 22  
VALUE STACK (TOP AT 133)  
C(GTR 48 49 |50)

AN EXPRESSION ENTRY IS MADE..WHILE A GTR B DO  
OPERANDS OF 48 AND 49 AND A RESULT OF 50.

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
EXCON N  
EXTYPE BOOL  
EXADD 0  
EXVAL# 50

B+23 0 |XIT | 137 | 00040025 |

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 23  
CONT STK <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
BLOCK# 2  
ENTRY 37

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
EXCON N  
EXTYPE BOOL  
EXADD 0  
EXVAL# 50



# TABLE V (CON'T)

B+24 0 IXIT . | 126 | 0004001A |

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 24  
 CONT STK <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
 BLOCK# 12 26  
 ENTRY 37 26  
 BLOCK #2 HAS GENERATED TWO IMMEDIATE SUCCESSOR BLOCKS.

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
 EXCON N  
 EXTYPE R00L  
 EXADD 0  
 EXVAL# 50

B+25 0 IBSC | 10 | 002F0000 |

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 3 IS:  
 C(48:49,\*,50) A(2,\*,49) A(1,\*,48) A(4,47,47) A(3,46,46)

CURRENT POOL FOR BOTH BLOCK  
 #3 AND #4 HAS BEEN SET TO THE  
 OUTPUT POOL FROM BLOCK #2.

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 4 IS:  
 C(48:49,\*,50) A(2,\*,49) A(1,\*,48) A(4,47,47) A(3,46,46)

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 25  
 ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
 ADDRESS 3 4 1 2 0  
 ADDTYPE INT INT INT INT GTR  
 ADDVAL 46 47 48 49 50  
 ADDCON 1 2 0 0 0

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
 (EMPTY)

CONTROL AT 26, FROM BLK 2 TO BLK 4 (PASS 1)  
 C+26 0 IENT IC160 | 0003803C |

CFA PASSES CONTROL FROM BLOCK #2 TO  
 BLOCK #4.

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 26

INPUT POOL FOR BLOCK# 4 IS:  
 C(48:49,\*,50) A(2,\*,49) A(1,\*,48) A(4,47,47) A(3,46,46)

INPUT POOL FOR BLOCK #4 IS  
 FOUR SIMPLE VARIABLES AND  
 ONE EXPRESSION.

ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
 ADDRESS 3 4 1 2  
 ADDTYPE INT INT INT INT  
 ADDVAL 46 47 48 49  
 ADDCON 1 2 0 0

VALUE STACK (TOP AT 133)  
 C(GTR 48 49 | 50)

CONT STK <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
 BLOCK# 2  
 ENTRY 37

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
 (EMPTY)

B+27 0 ILOC | 12 | 00020002 |

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 27  
 EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
 EXCON N  
 EXTYPE LOC  
 EXADD 2  
 EXVAL# 0

B+28 0 ILOC | 13 | 00020003 |

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 28  
 EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
 EXCON N  
 EXTYPE LOC  
 EXADD 2  
 EXVAL# 0



TABLE V (CON'T)

B+29 0 ILOD I 10 I 00270000 I

```

***T A B L E D U M P*** AT LOCATION = 29
EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
EXCON      N      Y
EXTYPE     LOC    INT
EXADD      2      1
EXVAL#     0      46

```

B+30 0 ILOC I 14 I 00020004 I

```

***T A B L E D U M P*** AT LOCATION = 30
EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
EXCON      N      Y      N
EXTYPE     LOC    INT    LOC
EXADD      2      1      4
EXVAL#     0      46     0

```

B+31 0 ILOD I 10 I 00270000 I

```

***T A B L E D U M P*** AT LOCATION = 31
EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
EXCON      N      Y      Y
EXTYPE     LOC    INT    INT
EXADD      2      1      2
EXVAL#     0      46     47

```

B+32 0 IADD I 10 I 00100000 I

```

***T A B L E D U M P*** AT LOCATION = 32
CONSTANTS <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
CONSYM     2      3      5
CONTYPE    INT    INT    INT
CONVAL     46     47     52
CONINT     2      3      5

```

NEW EXPRESSION ADDED  
B := C + D;

VALUE STACK (TOP AT 139)  
C(ADD 46 47 52) C(GTR 48 49 50)

```

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
EXCON      N      Y
EXTYPE     LOC    INT
EXADD      2      3
EXVAL#     0      52

```

B+33 0 ISTD I 10 I 00280000 I

```

***T A B L E D U M P*** AT LOCATION = 33
ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
ADDRESS   3      4      1      2      2
ADDTYPE    INT    INT    INT    INT    INT
ADDVAL     46     47     48     49     52
ADDCON     1      2      0      0      3

```

```

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
EXCON      Y
EXTYPE     INT
EXADD      3
EXVAL#     52

```

B+34 0 IDEL I 10 I 002A0000 I

```

***T A B L E D U M P*** AT LOCATION = 34
EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
(EMPTY)

```

B+35 0 IXIT I 17 I 00040011 I



TABLE V (CON'T)

```

***T A B L E   D U M P*** AT LOCATION = 35
CONT STK <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
BLOCK#  | 2 | 4 |
ENTRY   | 37 | 17 |

```

```


EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
(EMPTY)

```

```

B+36 0 |BRS      | 10 | 002E0000 |

```

BEFORE THE MEET OPERATION:  MEET OPERATION IS NOW PERFORMED USING THE FIRST BLOCK ON THE CONTROL LIST

OUTPUT POOL FROM BLOCK# 4 IS:  
C(48:49,\*,50) C(46:47,\*,52) A(2,52,52) A(1,\*,48) A(4,47,47) A(3,46,46)

CURRENT POOL FOR BLOCK# 2 IS:  
A(2,\*,49) A(1,\*,48) A(4,47,47) A(3,46,46)

CLASS OCCURENCE LIST:

ADDR	CLASS NUMBER	NUMBER OF OCCURRENCES
1	49	1
2	48	1
3	47	1
4	46	1

FOUR CLASS NUMBERS  
ARE IN THE OCCURENCE  
LIST

AFTER PROCESSING SIMPLE VARIABLES:

CLASS OCCURENCE LIST:  
(EMPTY)

SIMPLE VARIABLES HAVE EXHAUSTED THE  
CLASS OCCURENCE LIST

CLASS REFINEMENT TABLE:

LOC	NEW VALUE#	CURRENT VALUE#	INPUT VALUE#
1	53	49	52
2	48	48	48
3	47	47	47
4	46	46	46

NEW CLASS NUMBERS ASSIGNED  
ONLY IF CLASS NUMBERS DIFFER

AFTER PROCESSING ALL EXPRESSIONS:

CLASS OCCURENCE LIST:  
(EMPTY)

CLASS REFINEMENT TABLE:

LOC	NEW VALUE#	CURRENT VALUE#	INPUT VALUE#
1	53	49	52
2	48	48	48
3	47	47	47
4	46	46	46

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

BLOCK #2 WILL NOT BE PROCESSED  
BECAUSE ITS CURRENT POOL IS SAME

```

***T A B L E   D U M P*** AT LOCATION = 36
ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
ADDRESS  | 3 | 4 | 1 | 2 | 2 | 0 | 0 |
ADDTYPE  | INT | INT | INT | INT | INT | ADD | GTR |
ADDVAL   | 46 | 47 | 48 | 49 | 52 | 52 | 50 |
ADDCON   | 1 | 2 | 0 | 0 | 3 | 0 | 0 |

```

```

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>
(EMPTY)

```

CONTROL AT 37, FROM BLK 2 TO BLK 3 (PASS 1)  
C+37 0 |ENT |C|55 | 00038037 |

CFA PASSES CONTROL FROM BLOCK #2  
TO BLOCK #3.

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 37

INPUT POOL FOR BLOCK# 3 IS:  
C(48:49,\*,50) A(2,\*,49) A(1,\*,48) A(4,47,47) A(3,46,46)





TABLE V (CON'T)

ADDR	TABL	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
ADDRESS		3	4	1	2										
ADDTYPE		INT	INT	INT	INT										
ADDVAL		46	47	48	49										
ADDCON		1	2	0	0										

VALUE STACK (TOP AT 133)  
C(GTR 48 49 150)

CONT STK	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
(EMPTY)														

EXEC STAC	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
(EMPTY)														

B+38 0 |LOC | 15 | 00020005 |

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 38

ADDR	TABL	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
ADDRESS		3	4	1	2	5									
ADDTYPE		INT	INT	INT	INT	NULL									
ADDVAL		46	47	48	49	0									
ADDCON		1	2	0	0	0									

EXEC STAC	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
EXCON	N													
EXTYPE	LOC													
EXADD	5													
EXVAL#	0													

B+39 0 |LOC | 14 | 00020004 |

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 39

EXEC STAC	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
EXCON	N	N												
EXTYPE	LOC	LOC												
EXADD	5	4												
EXVAL#	0	0												

B+40 0 |LOD | 10 | 00270000 |

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 40

EXEC STAC	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
EXCON	N	Y												
EXTYPE	LOC	INT												
EXADD	5	2												
EXVAL#	0	47												

B+41 0 |LOC | 13 | 00020003 |

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 41

EXEC STAC	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
EXCON	N	Y	N											
EXTYPE	LOC	INT	LOC											
EXADD	5	2	3											
EXVAL#	0	47	0											

B+42 0 |LOD | 10 | 00270000 |

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 42

EXEC STAC	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
EXCON	N	Y	Y											
EXTYPE	LOC	INT	INT											
EXADD	5	2	1											
EXVAL#	0	47	46											

B+43 0 |ADD | 10 | 00100000 |

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 43  
VALUE STACK (TOP AT 139)  
C(ADD 47 46 152) C(GTR 48 49 150)



TABLE V (CON'T)

EXEC STAC	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
EXCON	N	Y												
EXTYPE	LOC	INT												
EXADD	5	3												
EXVAL#	0	52												

B+44 0 ISTO I 10 I 00280000 I

***T A B L E	D U M P***	AT LOCATION = 44												
ADDR TABL	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
ADDRESS	3	4	1	2	5									
ADDTYPE	INT	INT	INT	INT	INT									
ADDVAL	46	47	48	49	52									
ADDCON	1	2	0	0	3									

EXEC STAC	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
EXCON	Y													
EXTYPE	INT													
EXADD	3													
EXVAL#	52													

B+45 0 DEL I 10 I 002A3000 I

***T A B L E	D U M P***	AT LOCATION = 45												
EXEC STAC	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
(EMPTY)														

***T A B L E	D U M P***	AT LOCATION = 45												
ADDR TABL	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
ADDRESS	3	4	1	2	5	0	0							
ADDTYPE	INT	INT	INT	INT	INT	ADD	GTR							
ADDVAL	46	47	48	49	52	52	50							
ADDCON	1	2	0	0	3	0	0							

# FINAL OPTIMIZATION RESULTS

#####

OPTIMIZATION AT 32	←	AT LOCATION 32 AN EXPRESSION WAS
(47,46 32,52)		DETECTED AS A CONSTANT PROPAGATION
OPTIMIZATION AT 43	←	AT LOCATION 43 AN EXPRESSION IS IDENTICAL
(47,46 32,52)		TO AN EXPRESSION AT LOCATION 32



TABLE V (CON'T)

**** FORMATTED INTERMEDIATE CODE DUMP ****					OPTIMIZATION	
LOC	OFF	OP CODE	CNS ADR	RAW CODE	TYPE	
SET			/ETC			
*****						
0	0	ENT	46	0003002E		
1	0	TOGGLE	257	00080101		
2	0	TOGGLE	258	00080102		
3	0	LOC	3	00020003		
4	0	INT	1	00050001		
5	0	STO	0	00280000		
6	0	DEL	0	002A0000		
7	0	LOC	4	00020004		
8	0	INT	2	00050002		
9	0	STO	0	00280000		
10	0	DEL	0	002A0000		
11	0	LOC	1	00020001		
12	0	INT	0	00050000		
13	0	STO	0	00290000		
14	0	LOC	2	00020002		
15	0	INT	0	00050000		
16	0	STO	0	00290000		
17	0	ENT	50	00030032		
18	0	LOC	1	00020001		
19	0	LOD	0	00270000		
20	0	LOC	2	00020002		
21	0	LOD	0	00270000		
22	0	GTR	1024	00210400		
23	0	XIT	37	00040025		
24	0	XIT	26	0004001A		
25	0	BSC	0	002F0000		
26	0	ENT	60	0003003C		
27	0	LOC	2	00020002		
28	0	LOC	3	00020003		
29	0	LOD	0	00270000		
30	0	LOC	4	00020004		
31	0	LOD	2	00270000		
32	0	ADD	C 1016	001083F8	CONSTANT PROPAGATION	
33	0	STO	0	00280000		
34	0	DEL	0	002A0000		
35	0	XIT	17	00040011		
36	0	BRS	0	002E0000		
37	0	ENT	55	00030037		
38	0	LOC	5	00020005		
39	0	LOC	4	00020004		
40	0	LOD	0	00270000		
41	0	LOC	3	00020003		
42	0	LOD	2	00270000		
43	0	ADD	C 1012	001083F4	CONSTANT PROPAGATION	
44	0	STO	0	00280000		
45	0	DEL	0	002A0000		
*****						

OPTIMIZATIONS ARE FLAGGED IN  
IN THE C-FIELD

CONSTANT PROPAGATION

CONSTANT PROPAGATION



# TABLE V (CON'T)

FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

FINAL POOL FOR BLOCK# 1 IS: (NULL) ← ACTUALLY THE MOP FOR EACH BLOCK.

FINAL POOL FOR BLOCK# 2 IS:  
A(2,\*,53) A(1,\*,48) A(4,47,47) A(3,46,46)

FINAL POOL FOR BLOCK# 3 IS:  
C(48:49,\*,50) A(2,\*,49) A(1,\*,48) A(4,47,47) A(3,46,46)

FINAL POOL FOR BLOCK# 4 IS:  
C(48:49,\*,50) A(2,\*,49) A(1,\*,48) A(4,47,47) A(3,46,46)

BASIC BLOCK #1  
BEGINNING AT 0, ENDING AT 17  
BLOCK TRAVERSED 1 TIMES  
0 REFERENCES:  
BASIC BLOCK #2  
BEGINNING AT 17, ENDING AT 25  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
BASIC BLOCK #3  
BEGINNING AT 37, ENDING AT 45  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
BASIC BLOCK #4  
BEGINNING AT 26, ENDING AT 36  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:

BLOCK SUMMARY DATA:

BLOCK NUMBER	NUMBER OF PASSES
1	1
2	1
3	1
4	1

TOTAL NUMBER OF BLOCKS PROCESSED WAS 4  
USING THE STEEPEST DESCENT (MINIMUM CURRENT POOL) BLOCK SELECTION ALGORITHM.  
TIME FOR THE OPTIMIZATION WAS 0.233 SECONDS.

\*\*\*CODE SYNTHESIS FILTER IS COMPLETE\*\*\*





### III. TESTING OF BLOCK SELECTION METHOD

As previously mentioned, a principal objective of the implementation of Algorithm Q was to test block selection methods. Three classes of programs were analyzed using the CSF. A class of programs was developed by writing test programs that would encompass various typical program flows. This class is by far the more complex of the three test classes; however, it was felt that a "foreign" source for test programs should be sought to inject unbiased data. Using ALGOL-F programs written by students in an introductory ALGOL class, two samples were obtained. TABLES VI, VII and VIII are the synopses of the data compiled on these program sets. TABLES IX, X and XI give a statistical summary of the program classes while TABLE XII is a summary of all programs tested.

Lukasczyk [34] concluded that "steepest descent" outperformed LIFO, FIFO and the depth-first-search selection algorithm of Hecht and Ullman [23, 24]. The method used by Lukasczyk to heuristically arrive at steepest descent was to process the block with the minimum number of equivalence classes first. To observe the effects of increased complexity, the test programs were broken into five subclasses:

- 1) code size less than 200 words
- 2) code size from 200 to 400 words
- 3) code size from 400 to 600 words
- 4) code size from 600 to 800 words
- 5) code size greater than 800 words.



TABLES XIII, XIV, XV, XVI and XVII are the summary analyses of these subclasses.

Figure 9 is a graphical analysis of the five subclasses of program size tested. It shows that heuristic steepest descent results in a faster convergence rate for the global flow analysis algorithm given any of the code file sizes tested. The percentage of improvement between steepest descent and LIFO at each test level is:

<u>LEVEL</u>	<u>PERCENTAGE</u> <u>IMPROVEMENT</u>
less 200	12.5%
200 to 400	2.3%
400 to 600	4.4%
600 to 800	4.6%
greater 800	4.5%.

The improvement of steepest descent over LIFO is obvious and as Figure 9 shows, the other two methods tested are worse than LIFO in all cases. As can be seen, the percentage of improvement does not clearly justify the use of a specific selection method as program size increases. In fact, the supposition that the average number of blocks increases with code size is even suspect. Unfortunately, upon close examination the test data has deficiencies. The number of blocks processed for programs with a code file size from 0 to 400 words is biased from the generalized class of test programs. These programs were written to be as complex as possible and do not reflect a true "real world" program flow. The code file sizes analyzed are restricted to programs with less than 1000 words. This limitation resulted from the inability of the CSF to handle programs that include array structures. It is felt that the limited



sample size, coupled with the biased generalized programs has severely affected the test results.



TABLE VI

## DATA FOR STUDENT TEST #1

NUMBER		BLOCK SELECTION METHOD CHOSEN				
OPTIMS	CODE	BLOCKS	LIFO	FIFO	MIN POOL	MAX POOL
1	153	15	20	19	15	20
3	162	12	24	42	24	24
0	123	14	14	23	14	14
1	152	15	20	19	15	20
0	99	12	12	19	12	12
0	101	12	12	19	12	12
0	144	14	14	23	14	16
3	91	10	16	19	16	19
0	115	18	30	35	23	36
0	165	17	18	28	17	18
0	104	15	20	19	16	20
0	122	14	14	23	14	14
0	126	14	14	23	14	14
1	156	15	20	19	15	21
0	132	16	16	27	16	16
0	115	12	15	23	16	16
0	97	12	12	19	12	12
0	109	16	20	19	18	20
8	127	5	5	5	5	5
0	99	12	12	19	12	12
0	104	15	20	19	16	20
0	108	15	20	19	16	20
0	99	12	12	19	12	12
0	109	16	20	19	18	20
0	93	12	12	19	12	12
0	118	12	15	22	15	15





TABLE VI (CON'T)

0	96	12	12	19	12	12
0	94	12	12	19	12	12
0	101	12	12	19	12	12
0	111	16	20	19	18	20
0	71	12	12	19	12	12
0	71	12	12	19	12	12
0	81	16	23	21	19	23
0	99	12	12	19	12	12
0	93	12	12	19	12	12
0	107	16	23	21	17	23
0	107	16	23	21	17	23
0	107	16	23	21	17	23
0	87	16	20	19	17	20
5	728	9	13	13	13	13
0	102	15	20	19	16	20



TABLE VII

DATA FOR STUDENT TEST #2

NUMBER OPTIMS	BLOCK SELECTION METHOD CHOSEN					
	CODE	BLOCKS	LIFO	FIFO	MIN POOL	MAX POOL
2	161	24	53	51	38	68
4	513	9	13	13	13	13
5	540	9	14	14	14	14
5	728	9	13	13	13	13
6	719	11	18	18	18	18
9	859	12	19	22	19	20
4	544	9	13	13	13	13
4	979	9	13	13	13	13
7	348	31	66	70	74	76
6	665	11	15	15	14	15
4	812	13	18	21	16	16
7	708	10	16	20	15	15
5	625	9	16	16	16	16
5	582	10	14	14	14	14
4	928	11	15	15	15	15
6	781	14	20	22	19	20
5	693	12	16	16	16	16
11	307	25	42	48	44	46
9	286	18	40	43	38	42
10	296	24	42	56	43	64
6	418	16	26	30	24	26
8	584	17	28	33	26	28
11	926	19	31	38	29	34
10	629	21	36	43	32	38
10	289	25	43	56	45	48



TABLE VIII

## GENERALIZED STRUCTURE TEST

NUMBER		BLOCK SELECTION METHOD CHOSEN				
OPTIMS	CODE	BLOCKS	LIFO	FIFO	MIN POOL	MAX POOL
9	107	3	3	3	3	3
4	113	10	13	11	10	13
9	130	6	8	8	8	8
5	130	6	8	8	8	8
1	153	15	20	19	15	20
6	100	7	10	8	7	10
1	140	10	19	19	19	19
2	135	10	18	18	18	18
14	226	11	22	16	11	22
10	269	15	26	26	26	26
1	141	10	19	19	19	19
8	127	5	5	5	5	5
1	152	15	28	31	15	20
8	239	14	28	31	21	25
9	87	3	3	3	3	3
4	113	10	13	11	10	13
3	90	10	16	19	16	19
10	269	15	26	26	26	25
17	256	10	12	12	12	12
10	281	20	36	38	34	38
10	242	15	25	26	23	27
18	216	8	10	10	10	10
18	259	11	15	15	15	15
17	256	10	12	12	12	12
10	215	14	20	23	20	23
17	213	7	7	7	7	7
9	266	14	23	23	23	22
1	102	7	14	16	15	18



TABLE IX

## SUMMARY FOR STUDENT TEST # 1

BLOCK SELECTION METHOD	AVG BLOCKS PROCESSED	AVG PASSES PER BLOCK
LIFO	16.9	1.2
FIFO	20.9	1.5
MIN POOL	15.1	1.1
MAX POOL	17.2	1.2

AVERAGE CODE GENERATED: 126

AVERAGE OPTIMIZATIONS DETECTED: .3

AVERAGE BLOCKS: 13.8





TABLE X

## SUMMARY FOR STUDENT TEST # 2

BLOCK SELECTION METHOD	AVG BLOCKS PROCESSED	AVG PASSES PER BLOCK
LIFO	24.9	1.6
FIFO	27.4	1.8
MIN POOL	24.0	1.6
MAX POOL	27.2	1.7

AVERAGE CODE GENERATED: 610

AVERAGE OPTIMIZATIONS DETECTED: 6.4

AVERAGE BLOCKS: 14.7



TABLE XI

## SUMMARY FOR GENERALIZED STRUCTURE TEST

BLOCK SELECTION METHOD	AVG BLOCKS PROCESSED	AVG PASSES PER BLOCK
LIFO	16.8	1.5
FIFO	16.9	1.5
MIN POOL	15.4	1.4
MAX POOL	16.4	1.5

AVERAGE CODE GENERATED: 185

AVERAGE OPTIMIZATIONS DETECTED: 8.3

AVERAGE BLOCKS: 10.8



TABLE XII

SUMMARY FOR ALL TESTS

BLOCK SELECTION METHOD	AVG BLOCKS PROCESSED	AVG PASSES PER BLOCK
LIFO	19.0	1.4
FIFO	21.5	1.6
MIN POOL	17.5	1.3
MAX POOL	19.6	1.4

AVERAGE CODE GENERATED: 270

AVERAGE OPTIMIZATIONS DETECTED: 4.2

AVERAGE BLOCKS: 13.2



TABLE XIII

ANALYSIS OF CODE SIZE LESS THAN 200 WORDS

BLOCK SELECTION METHOD	AVG BLOCKS PROCESSED	AVG PASSES PER BLOCK
LIFO	16.8	1.3
FIFO	19.8	1.5
MIN POOL	14.9	1.2
MAX POOL	17.4	1.3

AVERAGE CODE GENERATED: 115

AVERAGE OPTIMIZATIONS DETECTED: 1.1

AVERAGE BLOCKS: 12.9





TABLE XIV

ANALYSIS OF CODE SIZE FROM 200 TO 400 WORDS

BLOCK SELECTION METHOD	AVG BLOCKS PROCESSED	AVG PASSES PER BLOCK
LIFO	27.5	1.6
FIFO	29.9	1.7
MIN POOL	26.9	1.6
MAX POOL	28.9	1.6

AVERAGE CODE GENERATED: 263

AVERAGE OPTIMIZATIONS DETECTED: 11.9

AVERAGE BLOCKS: 15.9



TABLE XV

ANALYSIS OF CODE SIZE FROM 400 TO 600 WORDS

BLOCK SELECTION METHOD	AVG BLOCKS PROCESSED	AVG PASSES PER BLOCK
LIFO	18.9	1.5
FIFO	20.7	1.6
MIN POOL	18.1	1.5
MAX POOL	18.9	1.5

AVERAGE CODE GENERATED: 524

AVERAGE OPTIMIZATIONS DETECTED: 5.5

AVERAGE BLOCKS: 12.2



TABLE XVI

ANALYSIS OF CODE SIZE FROM 600 TO 800 WORDS

BLOCK SELECTION METHOD	AVG BLOCKS PROCESSED	AVG PASSES PER BLOCK
LIFO	17.8	1.5
FIFO	19.1	1.6
MIN POOL	17.0	1.5
MAX POOL	17.9	1.5

AVERAGE CODE GENERATED: 694

AVERAGE OPTIMIZATIONS DETECTED: 6.1

AVERAGE BLOCKS: 11.7



TABLE XVII

ANALYSIS OF CODE FILE SIZE GREATER THAN 800 WORDS

BLOCK SELECTION METHOD	AVG BLOCKS PROCESSED	AVG PASSES PER BLOCK
LIFO	19.2	1.5
FIFO	21.8	1.7
MIN POOL	18.4	1.4
MAX POOL	19.6	1.5

AVERAGE CODE GENERATED: 901

AVERAGE OPTIMIZATIONS DETECTED: 6.4

AVERAGE BLOCKS: 12.8





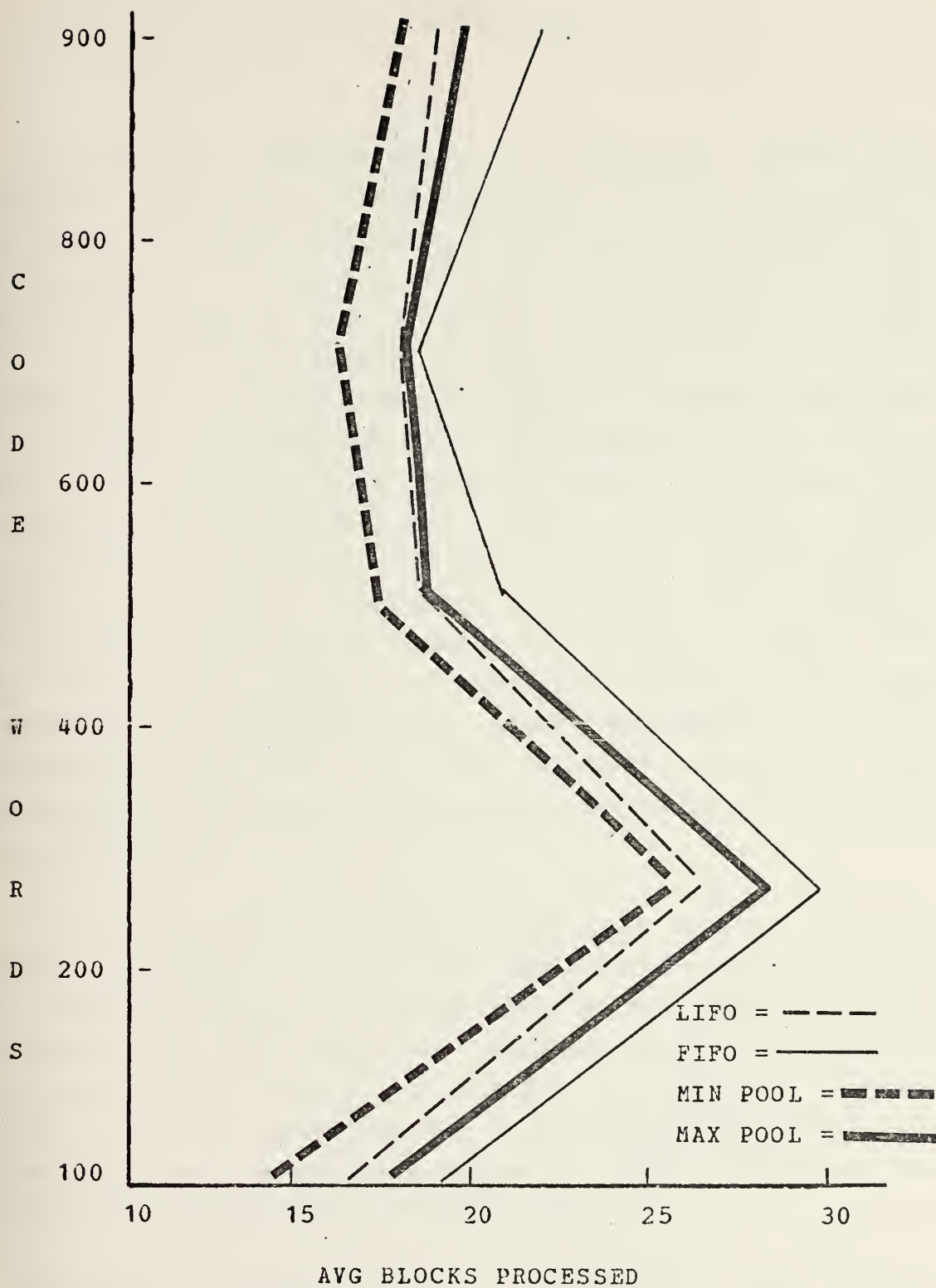


FIGURE 9  
RELATIONSHIP OF CODE SIZE AND BLOCKS PROCESSED



#### IV. CONCLUSIONS

A complete implementation of Kildall's global flow analysis algorithm has been presented. However, testing of convergence rates by different block selection methods was inconclusive. It is felt that the primary reason for this was due to a lack of test programs. Unfortunately, programs are available but cannot be used because of current CSF limitations. The inability to handle subscripted variables is a primary limitation. If this capability were added to the CSF, not only would a greater amount of small test programs be available, but, more importantly, larger programs could be tested. Kildall [29] has implemented a method of handling subscripts in a similar CSF optimizer. This method assigns class numbers to an array element only when the exact element is known and assumes a worst case assumption by nullifying all array assignments when an assignment is made to an unknown element. The implementation of this method in the CSF would be relatively straightforward.

Of the four block selection algorithms tested, steepest descent was shown to result in a faster convergence rate for the global flow analysis algorithm in all test samples. Unfortunately, LIFO was the easiest to implement and its small performance loss in relation to steepest descent cannot really be justified due to the additional coding and longer run time for steepest descent. FIFO performed the poorest of all selection methods tested.

In summary, steepest descent resulted in the fastest convergence rate for the global flow analysis algorithm, but statistically the improvement shown is not significant. Except for the possible rejection of FIFO, no real



conclusions as to which block selection method is better can be substantiated.

Live expression analysis is an area of interest that was not approached in this version of the CSF. Kildall [28, 30] has shown that a backward pass through the program to detect live expressions would allow the use of the live expression pool at each node as the initial current pool instead of the identity operand for the defined meet operator. Hand analysis using this method show a marked improvement in the convergence rate of the algorithm. An effort to implement a live expression analysis into the CSF should be attempted since it would considerably improve the convergence rate of the algorithm.



# COMPUTER OUTPUT

CARD	BL	SYL	COMMENT
1	0	0	TEST PROGRAM # 1 ;
2	0	0	\$EXECUTE
3	1	1	BEGIN LOCAL A,B,C,D,E,F,G,H,I,J;
4	1	1	A := 10;
5	1	5	B := 3;
6	1	9	C := 20;
7	1	13	D := 5;
8	1	17	E := 20;
9	1	21	F := 10;
10	1	25	WHILE A GTR B DO
11	1	35	BEGIN
12	1	35	WHILE C GTR D DO
13	2	45	BEGIN
14	2	45	WHILE E GTR F DO
15	3	55	BEGIN
16	3	55	B := B + 1;
17	4	62	F := F + 1;
18	4	69	D := D + 1;
19	4	76	END;
20	3	79	B := B + 1;
21	3	86	D := D + 1;
22	3	93	E := B;
23	3	98	H := E * C;
24	3	106	END;
25	2	109	B := B + 1;
26	2	116	E := C;
27	2	121	J := C * B;
28	2	129	I := B * C;
29	2	137	END;
30	1	140	END
31	1	140	EOF

CODE FILE COPIED (140 WORDS)  
 CONSTANT TABLE COPIED (10 WORDS)  
 2 RECORDS WRITTEN INTO FILE 1  
 END OF COMPILATION FEBRUARY 9, 1975. CLOCK TIME = 18:9:7.40.

31 CARDS WERE READ.  
 NO ERRORS WERE DETECTED.

SET UP TIME 0:0:0.04.  
 ACTUAL COMPILATION TIME 0:0:0.80.  
 CLEAN-UP TIME AT END 0:0:0.01.  
 TOTAL TIME IN COMPILER 0:0:0.85.  
 COMPILATION RATE :2325 CARDS PER MINUTE.





\*\*\*CODE SYNTHESIS FILTER\*\*\*

BLOCK SELECTION METHOD WILL BE STEEPEST DESCENT (MINIMUM CURRENT POOL)

\*\*\*COMPLETE TABLE DUMP\*\*\*  
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

\*\*\*TABLE DUMP\*\*\* AT LOCATION = 0

	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
CONSTANTS														
CONSYM	10	3	20	5	1									
CONTYPE	INT	INT	INT	INT	INT									
CONVAL	141	142	143	144	145									
CONINT	10	3	20	5	1									

ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

VALUE STACK (TOP AT 127)

	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
CONT STK														
BLOCK#	0													
ENTRY	0													

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

\*\*\*\* FORMATTED INTERMEDIATE CODE DUMP \*\*\*\*

LOC	OFF	OP CODE	CNS	ADR	RAW CODE	OPTIMIZATION
SET			/ETC			TYPE

0	0	ENT	C	141	0003808D	
1	0	TOGGLE		257	00080101	
2	0	LOC		1	00020001	
3	0	INT		1	00050001	
4	0	STO		0	00280000	
5	0	DEL		0	002A0000	
6	0	LOC		2	00020002	
7	0	INT		2	00050002	
8	0	STO		0	00280000	
9	0	DEL		0	002A0000	
10	0	LOC		3	00020003	
11	0	INT		3	00050003	
12	0	STO		0	00280000	
13	0	DEL		0	002A0000	
14	0	LOC		4	00020004	
15	0	INT		4	00050004	
16	0	STO		0	00280000	
17	0	DEL		0	002A0000	
18	0	LOC		5	00020005	
19	0	INT		3	00050003	
20	0	STO		0	00280000	
21	0	DEL		0	002A0000	
22	0	LOC		6	00020006	
23	0	INT		1	00050001	
24	0	STO		0	00280000	
25	0	DEL		0	002A0000	
26	0	ENT		1	00030001	
27	0	LOC		1	00020001	
28	0	LOC		0	00270000	
29	0	LOC		2	00020002	
30	0	LOC		0	00270000	
31	0	GTR		0	00210000	
32	0	XIT		140	0004000C	
33	0	XIT		35	00040023	
34	0	BSC		0	007F0000	
35	0	ENT		1	00030001	
36	0	ENT		1	00030001	
37	0	LOC		3	00020003	
38	0	LOC		0	00270000	
39	0	LOC		4	00020004	
40	0	LOC		0	00270000	
41	0	GTR		0	00210000	



42	0	XIT	109	0004006D
43	0	XIT	45	0004002D
44	0	BSC	0	002F0000
45	0	ENT	1	00030001
46	0	ENT	1	00030001
47	0	LOC	5	00020005
48	0	LOC	0	00270000
49	0	LOC	6	00020006
50	0	LOC	0	00270000
51	0	GTR	0	00210000
52	0	XIT	79	0004004F
53	0	XIT	55	00040037
54	0	BSC	0	002F0000
55	0	ENT	1	00030001
56	0	LOC	2	00020002
57	0	LOC	2	00020002
58	0	LOC	0	00270000
59	0	INT	5	00050005
60	0	ADD	0	00100000
61	0	STO	0	00280000
62	0	DEL	0	002A0000
63	0	LOC	6	00020006
64	0	LOC	6	00020006
65	0	LOC	5	00270000
66	0	INT	5	00050005
67	0	ADD	0	00100000
68	0	STO	0	00280000
69	0	DEL	0	002A0000
70	0	LOC	4	00020004
71	0	LOC	4	00020004
72	0	LOC	0	00270000
73	0	INT	5	00050005
74	0	ADD	0	00100000
75	0	STO	0	00280000
76	0	DEL	0	002A0000
77	0	XIT	46	0004002E
78	0	BRS	0	002E0000
79	0	ENT	1	00030001
80	0	LOC	2	00020002
81	0	LCC	2	00020002
82	0	LOC	0	00270000
83	0	INT	5	00050005
84	0	ADD	0	00100000
85	0	STO	0	00280000
86	0	DEL	0	002A0000
87	0	LOC	4	00020004
88	0	LOC	4	00020004
89	0	LOC	0	00270000
90	0	INT	5	00050005
91	0	ADD	0	00100000
92	0	STO	0	00280000
93	0	DEL	0	002A0000
94	0	LOC	5	00020005
95	0	LOC	2	00020002
96	0	LOC	0	00270000
97	0	STO	0	00280000
98	0	DEL	0	002A0000
99	0	LOC	8	00020008
100	0	LOC	5	00020005
101	0	LOC	0	00270000
102	0	LOC	3	00020003
103	0	LOC	0	00270000
104	0	MUL	0	00140000
105	0	STO	0	00280000
106	0	DEL	0	002A0000
107	0	XIT	36	00040024
108	0	BRS	0	002E0000
109	0	ENT	1	00030001
110	0	LOC	2	00020002
111	0	LOC	2	00020002
112	0	LOC	0	00270000
113	0	INT	5	00050005
114	0	ADD	0	00100000
115	0	STO	0	00280000
116	0	DEL	0	002A0000
117	0	LOC	5	00020005
118	0	LOC	3	00020003
119	0	LOC	0	00270000
120	0	STO	0	00280000
121	0	DEL	0	002A0000
122	0	LOC	10	0002000A
123	0	LOC	3	00020003
124	0	LOC	0	00270000
125	0	LOC	2	00020002
126	0	LOC	0	00270000
127	0	MUL	0	00140000
128	0	STO	0	00280000
129	0	DEL	0	002A0000



130	0	LOC	2	00020009
131	0	LOC	2	00020002
132	0	LOC	0	00270000
133	0	LOC	3	00020003
134	0	LOC	0	00270000
135	0	MUL	0	00140000
136	0	STO	0	00280000
137	0	DEL	0	002A0000
138	0	XIT	26	0004001A
139	0	BRS	0	002E0000
140	0	ENT	1	00030001

\*\*\*\*\*

BASIC BLOCK #1  
 BEGINNING AT 0, ENDING AT 0  
 BLOCK TRAVERSED 0 TIMES  
 0 REFERENCES:

\*\*\*END OF COMPLETE TABLE DUMP\*\*\*

B+2	0	LOC	1	00020001
B+3	0	INT	1	00050001
B+4	0	STO	0	00280000
B+5	0	DEL	0	002A0000
B+6	0	LOC	2	00020002
B+7	0	INT	2	00050002
B+8	0	STO	0	00280000
B+9	0	DEL	0	002A0000
B+10	0	LOC	3	00020003
B+11	0	INT	3	00050003
B+12	0	STO	0	00280000
B+13	0	DEL	0	002A0000
B+14	0	LOC	4	00020004
B+15	0	INT	4	00050004
B+16	0	STO	0	00280000
B+17	0	DEL	0	002A0000
B+18	0	LOC	5	00020005
B+19	0	INT	3	00050003
B+20	0	STO	0	00280000
B+21	0	DEL	0	002A0000
B+22	0	LOC	6	00020006
B+23	0	INT	1	00050001
B+24	0	STO	0	00280000
B+25	0	DEL	0	002A0000
B+26	0	ENT	1	00030001

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 2 IS:  
 A(6,141,141) A(5,143,143) A(4,144,144) A(3,143,143) A(2,142,142)  
 A(1,141,141)

CONTROL AT 26, FROM BLK 1 TO BLK 2 (PASS 1)

C+26	0	ENT	C 145	00038091
B+27	0	LOC	1	00020001
B+28	0	LOC	0	00270000
B+29	0	LOC	2	00020002
B+30	0	LOC	0	00270000
B+31	0	GTR	0	00210000
B+32	0	XIT	140	0004008C
B+33	0	XIT	35	00040023
B+34	0	RSC	0	002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 3 IS:  
 C(141:142,\*,147) A(6,141,141) A(5,143,143) A(4,144,144) A(3,143,143)  
 A(2,142,142) A(1,141,141)

CONTROL AT 35, FROM BLK 2 TO BLK 3 (PASS 1)

C+35	0	ENT	C 150	00038096
B+36	0	ENT	1	00030001

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 4 IS:  
 C(141:142,\*,147) A(6,141,141) A(5,143,143) A(4,144,144) A(3,143,143)  
 A(2,142,142) A(1,141,141)

CONTROL AT 36, FROM BLK 3 TO BLK 4 (PASS 1)

C+36	0	ENT	C 155	00038098
B+37	0	LOC	3	00020003
B+38	0	LOC	0	00270000
B+39	0	LOC	4	00020004
B+40	0	LOC	0	00270000
B+41	0	GTR	0	00210000
B+42	0	XIT	109	0004006D
B+43	0	XIT	45	0004002D
B+44	0	BSC	0	002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 5 IS:  
 C(141:142,\*,147) C(143:144,\*,147) A(6,141,141) A(5,143,143) A(4,144,144)  
 A(3,143,143) A(2,142,142) A(1,141,141)





CONTROL AT 45, FROM BLK 4 TO BLK 5 (PASS 1)

C+45	0	ENT	C	160	000380A0
B+46	0	ENT		1	00030001

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 6 IS:  
 C(143:144,\*,147) C(141:142,\*,147) A(6,141,141) A(5,143,143) A(4,144,144)  
 A(3,143,143) A(2,142,142) A(1,141,141)

CONTROL AT 46, FROM BLK 5 TO BLK 6 (PASS 1)

C+46	0	ENT	C	165	000380A5
B+47	0	LOC		5	00020005
B+48	0	LOC		0	00270000
B+49	0	LOC		6	00020006
B+50	0	LOC		0	00270000
B+51	0	GTR		0	00210000
B+52	0	XIT		79	0004004F
B+53	0	XIT		55	00040037
B+54	0	BSC		0	002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 7 IS:  
 C(141:142,\*,147) C(143:144,\*,147) C(143:141,\*,147) A(6,141,141)  
 A(5,143,143) A(4,144,144) A(3,143,143) A(2,142,142) A(1,141,141)

CONTROL AT 55, FROM BLK 6 TO BLK 7 (PASS 1)

C+55	0	ENT	C	170	000380AA
B+56	0	LOC		2	00020002
B+57	0	LOC		2	00020002
B+58	0	LOC		0	00270000
B+59	0	INT		5	00050005
B+60	0	ADD		0	00100000
B+61	0	STO		0	00280000
B+62	0	DEL		0	002A0000
B+63	0	LOC		6	00020006
B+64	0	LOC		6	00020006
B+65	0	LOC		0	00270000
B+66	0	INT		5	00050005
B+67	0	ADD		0	00100000
B+68	0	STO		0	00280000
B+69	0	DEL		0	002A0000
B+70	0	LOC		4	00020004
B+71	0	LOC		4	00020004
B+72	0	LOC		0	00270000
B+73	0	INT		5	00050005
B+74	0	ADD		0	00100000
B+75	0	STO		0	00280000
B+76	0	DEL		0	002A0000
B+77	0	XIT		46	0004002E
B+78	0	BRS		0	002E0000

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 7 IS:  
 C(143:141,\*,147) C(143:144,\*,147) C(141:142,\*,147) C(142:145,\*,151)  
 C(141:145,\*,153) C(144:145,\*,155) A(4,155,155) A(6,153,153) A(2,151,151)  
 A(5,143,143) A(3,143,143) A(1,141,141)

CURRENT POOL FOR BLOCK# 6 IS:  
 C(143:144,\*,147) C(141:142,\*,147) A(6,141,141) A(5,143,143) A(4,144,144)  
 A(3,143,143) A(2,142,142) A(1,141,141)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 6 IS:  
 A(6,\*,156) A(5,143,143) A(4,\*,157) A(3,143,143) A(2,\*,158) A(1,141,141)

CONTROL AT 46, FROM BLK 7 TO BLK 6 (PASS 2)

C+46	0	ENT	C	165	000380A5
B+47	0	LOC		5	00020005
B+48	0	LOC		0	00270000
B+49	0	LOC		6	00020006
B+50	0	LOC		0	00270000
B+51	0	GTR	C	1004	002183EC
B+52	0	XIT		79	0004004F
B+53	0	XIT		55	00040037
B+54	0	BSC		0	002F0000

CONSTANT PROPAGATION

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 8 IS:  
 C(143:156,\*,159) A(6,\*,156) A(5,143,143) A(4,\*,157) A(3,143,143)  
 A(2,\*,158) A(1,141,141)

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 6 IS:  
 C(143:156,\*,159) A(6,\*,156) A(5,143,143) A(4,\*,157) A(3,143,143)  
 A(2,\*,158) A(1,141,141)





CURRENT POOL FOR BLOCK# 7 IS:

C(141:142,\*,147) C(143:144,\*,147) C(143:141,\*,147) A(6,141,141)  
A(5,143,143) A(4,144,144) A(3,143,143) A(2,142,142) A(1,141,141)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 7 IS:

A(6,\*,160) A(5,143,143) A(4,\*,161) A(3,143,143) A(2,\*,162) A(1,141,141)

CONTROL AT 55, FROM BLK 6 TO BLK 7 (PASS 2)

C+55	0	ENT	C 170	000380AA	
B+56	0	LOC	2	00020002	
B+57	0	LOC	2	00020002	
B+58	0	LOC	0	00270000	
B+59	0	INT	5	00050005	
B+60	0	ADD	C 994	001083E2	CONSTANT PROPAGATION
B+61	0	STO	0	00280000	
B+62	0	DEL	0	002A0000	
B+63	0	LOC	6	00020006	
B+64	0	LOC	6	00020006	
B+65	0	LOC	0	00270000	
B+66	0	INT	5	00050005	
B+67	0	ADD	C 990	001083DE	CONSTANT PROPAGATION
B+68	0	STO	0	00280000	
B+69	0	DEL	0	002A0000	
B+70	0	LOC	4	00020004	
B+71	0	LOC	4	00020004	
B+72	0	LOC	0	00270000	
B+73	0	INT	5	00050005	
B+74	0	ADD	C 986	001083DA	CONSTANT PROPAGATION
B+75	0	STO	0	00280000	
B+76	0	DEL	0	002A0000	
B+77	0	XIT	46	0004002E	
B+78	0	BRS	0	002E0000	

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 7 IS:

C(162:145,\*,163) C(160:145,\*,164) C(161:145,\*,165) A(4,\*,155) A(6,\*,164)  
A(2,\*,163) A(5,143,143) A(3,143,143) A(1,141,141)

CURRENT POOL FOR BLOCK# 6 IS:

A(6,\*,156) A(5,143,143) A(4,\*,157) A(3,143,143) A(2,\*,158) A(1,141,141)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

CONTROL AT 79, FROM BLK 6 TO BLK 8 (PASS 1)

C+79	0	ENT	C 175	000380AF
B+80	0	LOC	2	00020002
B+81	0	LOC	2	00020002
B+82	0	LOC	0	00270000
B+83	0	INT	5	00050005
B+84	0	ADD	0	00100000
B+85	0	STO	0	00280000
B+86	0	DEL	0	002A0000
B+87	0	LOC	4	00020004
B+88	0	LOC	4	00020004
B+89	0	LOC	0	00270000
B+90	0	INT	5	00050005
B+91	0	ADD	0	00100000
B+92	0	STO	0	00280000
B+93	0	DEL	0	002A0000
B+94	0	LOC	5	00020005
B+95	0	LOC	2	00020002
B+96	0	LOC	0	00270000
B+97	0	STO	0	00280000
B+98	0	DEL	0	002A0000
B+99	0	LOC	8	00020008
B+100	0	LOC	5	00020005
B+101	0	LOC	0	00270000
B+102	0	LOC	3	00020003
B+103	0	LOC	0	00270000
B+104	0	MUL	0	00140000
B+105	0	STO	0	00280000
B+106	0	DEL	0	002A0000
B+107	0	XIT	36	00040024
B+108	0	BRS	0	002E0000

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 8 IS:

C(143:156,\*,159) C(158:145,\*,169) C(157:145,\*,170) C(169:143,\*,171)  
C(143:156,\*,171) A(8,\*,171) A(5,\*,169) A(4,\*,170) A(2,\*,169) A(6,\*,156)  
A(3,143,143) A(1,141,141)



CURRENT POOL FOR BLOCK# 4 IS:  
 C(141:142,\*,147) A(6,141,141) A(5,143,143) A(4,144,144) A(3,143,143)  
 A(2,142,142) A(1,141,141)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 4 IS:  
 A(6,\*,172) A(5,\*,173) A(4,\*,174) A(3,143,143) A(2,\*,175) A(1,141,141)

CONTROL AT 36, FROM BLK 8 TO BLK 4 (PASS 2)			
C+36	0	ENT	C 155 0003879B
B+37	0	LOC	3 00020003
B+38	0	LOD	0 00270000
B+39	0	LOC	4 00020004
B+40	0	LOD	0 00270000
B+41	0	GTR	C 1016 002183F8
B+42	0	XIT	109 0004006D
B+43	0	XIT	45 0004002D
B+44	0	BSC	0 002F0000

CONSTANT PROPAGATION

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 9 IS:  
 C(143:174,\*,176) A(6,\*,172) A(5,\*,173) A(4,\*,174) A(3,143,143)  
 A(2,\*,175) A(1,141,141)

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 4 IS:  
 C(143:174,\*,176) A(6,\*,172) A(5,\*,173) A(4,\*,174) A(3,143,143)  
 A(2,\*,175) A(1,141,141)

CURRENT POOL FOR BLOCK# 5 IS:  
 C(141:142,\*,147) C(143:144,\*,147) A(6,141,141) A(5,143,143) A(4,144,144)  
 A(3,143,143) A(2,142,142) A(1,141,141)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 5 IS:  
 A(6,\*,177) A(5,\*,178) A(4,\*,179) A(3,143,143) A(2,\*,180) A(1,141,141)

CONTROL AT 45, FROM BLK 4 TO BLK 5 (PASS 2)			
C+45	0	ENT	C 160 000380A0
B+46	0	ENT	C 165 000380A5

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 5 IS:  
 A(6,\*,177) A(5,\*,178) A(4,\*,179) A(3,143,143) A(2,\*,180) A(1,141,141)

CURRENT POOL FOR BLOCK# 6 IS:  
 A(6,\*,166) A(5,143,143) A(4,\*,167) A(3,143,143) A(2,\*,168) A(1,141,141)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 6 IS:  
 A(6,\*,181) A(5,\*,182) A(4,\*,183) A(3,143,143) A(2,\*,184) A(1,141,141)

CONTROL AT 46, FROM BLK 5 TO BLK 6 (PASS 3)			
C+46	0	ENT	C 165 000380A5
B+47	0	LOC	5 00020005
B+48	0	LOD	0 00270000
B+49	0	LOC	6 00020006
B+50	0	LOD	0 00270000
B+51	0	GTR	C 1004 002183EC
B+52	0	XIT	79 0004004F
B+53	0	XIT	55 00040037
B+54	0	BSC	0 002F0000

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 6 IS:  
 C(182:181,\*,185) A(6,\*,181) A(5,\*,182) A(4,\*,183) A(3,143,143)  
 A(2,\*,184) A(1,141,141)

CURRENT POOL FOR BLOCK# 8 IS:  
 C(143:156,\*,159) A(6,\*,156) A(5,143,143) A(4,\*,157) A(3,143,143)  
 A(2,\*,158) A(1,141,141)



AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 8 IS:

A(6,\*,186) A(5,\*,187) A(4,\*,188) A(3,143,143) A(2,\*,189) A(1,141,141)

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 6 IS:

C(182:181,\*,185) A(6,\*,181) A(5,\*,182) A(4,\*,183) A(3,143,143)  
A(2,\*,184) A(1,141,141)

CURRENT POOL FOR BLOCK# 7 IS:

A(6,\*,160) A(5,143,143) A(4,\*,161) A(3,143,143) A(2,\*,162) A(1,141,141)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 7 IS:

A(6,\*,190) A(5,\*,191) A(4,\*,192) A(3,143,143) A(2,\*,193) A(1,141,141)

CONTROL AT 55, FROM BLK 6 TO BLK 7 (PASS 3)

C+55	0	ENT	C 170	000380AA
B+56	0	LOC	2	00020002
B+57	0	LOC	2	00020002
B+58	0	LOC	0	00270000
B+59	0	INT	5	00050005
B+60	0	ADD	C 994	001083E2
B+61	0	STO	0	00280000
B+62	0	DEL	0	002A0000
B+63	0	LOC	6	00020006
B+64	0	LOC	6	00020006
B+65	0	LOC	0	00270000
B+66	0	INT	5	00050005
B+67	0	ADD	C 990	001083DE
B+68	0	STO	0	00280000
B+69	0	DEL	0	002A0000
B+70	0	LOC	4	00020004
B+71	0	LOC	4	00020004
B+72	0	LOC	0	00270000
B+73	0	INT	5	00050005
B+74	0	ADD	C 986	001083DA
B+75	0	STO	0	00280000
B+76	0	DEL	0	002A0000
B+77	0	XIT	46	0004002E
B+78	0	8RS	0	002E0000

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 7 IS:

C(193:145,\*,194) C(190:145,\*,195) C(192:145,\*,196) A(4,\*,196) A(6,\*,195)  
A(2,\*,194) A(5,\*,191) A(3,143,143) A(1,141,141)

CURRENT POOL FOR BLOCK# 6 IS:

A(6,\*,181) A(5,\*,182) A(4,\*,183) A(3,143,143) A(2,\*,184) A(1,141,141)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

CONTROL AT 79, FROM BLK 6 TO BLK 8 (PASS 2)

C+79	0	ENT	C 175	000380AF
B+80	0	LOC	2	00020002
B+81	0	LOC	2	00020002
B+82	0	LOC	0	00270000
B+83	0	INT	5	00050005
B+84	0	ADD	C 980	001083D4
B+85	0	STO	0	00280000
B+86	0	DEL	0	002A0000
B+87	0	LOC	4	00020004
B+88	0	LOC	4	00020004
B+89	0	LOC	0	00270000
B+90	0	INT	5	00050005
B+91	0	ADD	C 976	001033D0
B+92	0	STO	0	00280000
B+93	0	DEL	0	002A0000
B+94	0	LOC	5	00020005
B+95	0	LOC	2	00020002
B+96	0	LOC	0	00270000
B+97	0	STO	0	00280000
B+98	0	DEL	0	002A0000
B+99	0	LOC	8	00020008
B+100	0	LOC	5	00020005
B+101	0	LOC	0	00270000
B+102	0	LOC	3	00020003
B+103	0	LOC	0	00270000





B+104	0	MUL	C	9/2	001483CC
B+105	0	STO		0	00280000
B+106	0	DEL		0	002A0000
B+107	0	XIT		36	00040024
B+108	0	BRS		0	002E0000

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 8 IS:

C(189:145,\*,201) C(188:145,\*,202) C(201:143,\*,203) A(8,\*,203) A(5,\*,201)  
A(4,\*,202) A(2,\*,201) A(6,\*,186) A(3,143,143) A(1,141,141)

CURRENT POOL FOR BLOCK# 4 IS:

A(6,\*,172) A(5,\*,173) A(4,\*,174) A(3,143,143) A(2,\*,175) A(1,141,141)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

CONTROL AT 109, FROM BLK 4 TO BLK 9 (PASS 1)					
C+109	0	ENT	C	180	00038084
B+110	0	LOC		2	00020002
B+111	0	LOC		2	00020002
B+112	0	LOC		0	00270000
B+113	0	INT		5	00050005
B+114	0	ADD		0	00100000
B+115	0	STO		0	00280000
B+116	0	DEL		0	002A0000
B+117	0	LOC		5	00020005
B+118	0	LOC		3	00020003
B+119	0	LOC		0	00270000
B+120	0	STO		0	00280000
B+121	0	DEL		0	002A0000
B+122	0	LOC		10	0002000A
B+123	0	LOC		3	00020003
B+124	0	LOC		0	00270000
B+125	0	LOC		2	00020002
B+126	0	LOC		0	00270000
B+127	0	MUL		0	00140000
B+128	0	STO		0	00280000
B+129	0	DEL		0	002A0000
B+130	0	LOC		9	00020009
B+131	0	LOC		2	00020002
B+132	0	LOC		0	00270000
B+133	0	LOC		3	00020003
B+134	0	LOC		0	00270000
B+135	0	MUL		0	00140000
B+136	0	STO		0	00280000
B+137	0	DEL		0	002A0000
B+138	0	XIT		26	0004001A
B+139	0	BRS		0	002E0000

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 9 IS:

C(143:174,\*,176) C(175:145,\*,208) C(143:208,\*,209) A(9,\*,209) A(10,\*,209)  
A(5,143,143) A(2,\*,208) A(6,\*,172) A(4,\*,174) A(3,143,143) A(1,141,141)

CURRENT POOL FOR BLOCK# 2 IS:

A(6,141,141) A(5,143,143) A(4,144,144) A(3,143,143) A(2,142,142)  
A(1,141,141)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 2 IS:

A(6,\*,210) A(5,143,143) A(4,\*,211) A(3,143,143) A(2,\*,212) A(1,141,141)

CONTROL AT 26, FROM BLK 9 TO BLK 2 (PASS 2)					
C+26	0	ENT	C	145	00038091
B+27	0	LOC		1	00020001
B+28	0	LOC		0	00270000
B+29	0	LOC		2	00020002
B+30	0	LOC		0	00270000
B+31	0	GTR	C	1024	00218400
B+32	0	XIT		140	0004008C
B+33	0	XIT		35	00040023
B+34	0	BSC		0	002F0000

CONSTANT PROPAGATION

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 10 IS:

C(141:212,\*,213) A(6,\*,210) A(5,143,143) A(4,\*,211) A(3,143,143)  
A(2,\*,212) A(1,141,141)

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 2 IS:

C(141:212,\*,213) A(6,\*,210) A(5,143,143) A(4,\*,211) A(3,143,143)  
A(2,\*,212) A(1,141,141)





CURRENT POOL FOR BLOCK# 3 IS:  
 C(141:142,\*,147) A(6,141,141) A(5,143,143) A(4,144,144) A(3,143,143)  
 A(2,142,142) A(1,141,141)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 3 IS:  
 A(6,\*,214) A(5,143,143) A(4,\*,215) A(3,143,143) A(2,\*,216) A(1,141,141)

CONTROL AT 35, FROM BLK 2 TO BLK 3 (PASS 2)  
 C+35 0 ENT C150 00038096  
 B+36 0 ENT C155 00038098

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 3 IS:  
 A(6,\*,214) A(5,143,143) A(4,\*,215) A(3,143,143) A(2,\*,216) A(1,141,141)

CURRENT POOL FOR BLOCK# 4 IS:  
 A(6,\*,204) A(5,\*,205) A(4,\*,206) A(3,143,143) A(2,\*,207) A(1,141,141)

AFTER PERFORMING THE MEET OPERATION:  
 NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

CONTROL AT 140, FROM BLK 2 TO BLK 10 (PASS 1)  
 C+140 0 ENT C185 00038089

# FINAL OPTIMIZATION RESULTS

OPTIMIZATION AT 135  
 (143,208|127,209) COMM SUBEXP ELIM

```

**** FORMATTED INTERMEDIATE CODE DUMP ****
LOC OFF  OP CODE  CNS ADR  RAW CODE  OPTIMIZATION
SET      /ETC      TYPE
*****
0      0      ENT      141      00030080
1      0      TOGGLE   257      00080101
2      0      LOC      1       00020001
3      0      INT      1       00050001
4      0      STO      0       00280000
5      0      DEL      0       002A0000
6      0      LOC      2       00020002
7      0      INT      2       00050002
8      0      STO      0       00280000
9      0      DEL      0       002A0000
10     0      LOC      3       00020003
11     0      INT      3       00050003
12     0      STO      0       00280000
13     0      DEL      0       002A0000
14     0      LOC      4       00020004
15     0      INT      4       00050004
16     0      STO      0       00280000
17     0      DEL      0       002A0000
18     0      LOC      5       00020005
19     0      INT      3       00050003
20     0      STO      0       00280000
21     0      DEL      0       002A0000
22     0      LOC      6       00020006
23     0      INT      1       00050001
24     0      STO      0       00280000
25     0      DEL      0       002A0000
26     0      ENT      145     00030091
27     0      LOC      1       00020001
28     0      LOD      0       00270000
29     0      LOC      2       00020002
30     0      LOD      0       00270000
31     0      GTR      1024    00210400
32     0      XIT      140     0004008C
33     0      XIT      35      00040023
34     0      BSC      0       002F0000
35     0      ENT      150     00030096
36     0      ENT      155     00030098
37     0      LOC      3       00020003
38     0      LOD      0       00270000
39     0      LOC      4       00020004
40     0      LOD      0       00270000

```



41	U	GTR	1010	002103F8
42	0	XIT	109	00040060
43	0	XIT	45	00040020
44	0	BSC	0	002F0000
45	0	ENT	160	000300A0
46	0	ENT	165	000300A5
47	0	LOC	5	00020005
48	0	LOC	0	00270000
49	0	LOC	6	00020006
50	0	LOC	0	00270000
51	0	GTR	1004	002103EC
52	0	XIT	79	0004004F
53	0	XIT	55	00040037
54	0	BSC	0	002F0000
55	0	ENT	170	000300AA
56	0	LOC	2	00020002
57	0	LOC	2	00020002
58	0	LOC	0	00270000
59	0	INT	5	00050005
60	0	ADD	994	001003E2
61	0	STO	0	00280000
62	0	DEL	0	002A0000
63	0	LOC	6	00020006
64	0	LOC	6	00020006
65	0	LOC	0	00270000
66	0	INT	5	00050005
67	0	ADD	990	001003DE
68	0	STO	0	00280000
69	0	DEL	0	002A0000
70	0	LOC	4	00020004
71	0	LOC	4	00020004
72	0	LOC	0	00270000
73	0	INT	5	00050005
74	0	ADD	986	001003DA
75	0	STO	0	00280000
76	0	DEL	0	002A0000
77	0	XIT	46	0004002E
78	0	BRS	0	002E0000
79	0	ENT	175	000300AF
80	0	LOC	2	00020002
81	0	LOC	2	00020002
82	0	LOC	0	00270000
83	0	INT	5	00050005
84	0	ADD	980	001003D4
85	0	STO	0	00280000
86	0	DEL	0	002A0000
87	0	LOC	4	00020004
88	0	LOC	4	00020004
89	0	LOC	0	00270000
90	0	INT	5	00050005
91	0	ADD	976	001003D0
92	0	STO	0	00280000
93	0	DEL	0	002A0000
94	0	LOC	5	00020005
95	0	LOC	2	00020002
96	0	LOC	0	00270000
97	0	STO	0	00280000
98	0	DEL	0	002A0000
99	0	LOC	8	00020008
100	0	LOC	5	00020005
101	0	LOC	0	00270000
102	0	LOC	3	00020003
103	0	LOC	0	00270000
104	0	MUL	972	001403CC
105	0	STO	0	00280000
106	0	DEL	0	002A0000
107	0	XIT	36	00040024
108	0	BRS	0	002E0000
109	0	ENT	180	000300B4
110	0	LOC	2	00020002
111	0	LOC	2	00020002
112	0	LOC	0	00270000
113	0	INT	5	00050005
114	0	ADD	966	001003C6
115	0	STO	0	00280000
116	0	DEL	0	002A0000
117	0	LOC	5	00020005
118	0	LOC	3	00020003
119	0	LOC	0	00270000
120	0	STO	0	00280000
121	0	DEL	0	002A0000
122	0	LOC	10	0002000A
123	0	LOC	3	00020003
124	0	LOC	0	00270000
125	0	LOC	2	00020002
126	0	LOC	0	00270000
127	0	MUL	962	001403C2
128	0	STO	0	00280000



129	U	DEL	0	002A0000	
130	0	LOC	9	00020009	
131	0	LOC	2	00020002	
132	0	LOC	0	00270000	
133	0	LOC	3	00020003	
134	0	LOC	0	00270000	
135	0	MUL	C 958	001483BE	COMM SUBEXP ELIM
136	0	STO	0	00280000	
137	0	DEL	0	002A0000	
138	0	XIT	26	0004001A	
139	0	BRS	0	002E0000	
140	0	ENT	185	000300B9	

\*\*\*\*\*

FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

FINAL POOL FOR BLOCK# 1 IS:  
(NULL)

FINAL POOL FOR BLOCK# 2 IS:  
A(6,\*,210) A(5,143,143) A(4,\*,211) A(3,143,143) A(2,\*,212) A(1,141,141)

FINAL POOL FOR BLOCK# 3 IS:  
A(6,\*,214) A(5,143,143) A(4,\*,215) A(3,143,143) A(2,\*,216) A(1,141,141)

FINAL POOL FOR BLOCK# 4 IS:  
A(6,\*,217) A(5,\*,218) A(4,\*,219) A(3,143,143) A(2,\*,220) A(1,141,141)

FINAL POOL FOR BLOCK# 5 IS:  
A(6,\*,177) A(5,\*,178) A(4,\*,179) A(3,143,143) A(2,\*,180) A(1,141,141)

FINAL POOL FOR BLOCK# 6 IS:  
A(6,\*,197) A(5,\*,198) A(4,\*,199) A(3,143,143) A(2,\*,200) A(1,141,141)

FINAL POOL FOR BLOCK# 7 IS:  
A(6,\*,190) A(5,\*,191) A(4,\*,192) A(3,143,143) A(2,\*,193) A(1,141,141)

FINAL POOL FOR BLOCK# 8 IS:  
A(6,\*,186) A(5,\*,187) A(4,\*,188) A(3,143,143) A(2,\*,189) A(1,141,141)

FINAL POOL FOR BLOCK# 9 IS:  
C(143:174,\*,176) A(6,\*,172) A(5,\*,173) A(4,\*,174) A(3,143,143)  
A(2,\*,175) A(1,141,141)

FINAL POOL FOR BLOCK# 10 IS:  
C(141:212,\*,213) A(6,\*,210) A(5,143,143) A(4,\*,211) A(3,143,143)  
A(2,\*,212) A(1,141,141)



BASIC BLOCK #1  
 BEGINNING AT 0, ENDING AT 26  
 BLOCK TRAVERSED 1 TIMES  
 0 REFERENCES:

BASIC BLOCK #2  
 BEGINNING AT 26, ENDING AT 34  
 BLOCK TRAVERSED 2 TIMES  
 1 REFERENCES:

BASIC BLOCK #3  
 BEGINNING AT 35, ENDING AT 36  
 BLOCK TRAVERSED 2 TIMES  
 1 REFERENCES:

BASIC BLOCK #4  
 BEGINNING AT 36, ENDING AT 44  
 BLOCK TRAVERSED 2 TIMES  
 1 REFERENCES:

BASIC BLOCK #5  
 BEGINNING AT 45, ENDING AT 46  
 BLOCK TRAVERSED 2 TIMES  
 1 REFERENCES:

BASIC BLOCK #6  
 BEGINNING AT 46, ENDING AT 54  
 BLOCK TRAVERSED 3 TIMES  
 1 REFERENCES:

BASIC BLOCK #7  
 BEGINNING AT 55, ENDING AT 78  
 BLOCK TRAVERSED 3 TIMES  
 1 REFERENCES:

BASIC BLOCK #8  
 BEGINNING AT 79, ENDING AT 108  
 BLOCK TRAVERSED 2 TIMES  
 1 REFERENCES:

BASIC BLOCK #9  
 BEGINNING AT 109, ENDING AT 139  
 BLOCK TRAVERSED 1 TIMES  
 1 REFERENCES:

BASIC BLOCK #10  
 BEGINNING AT 140, ENDING AT 140  
 BLOCK TRAVERSED 1 TIMES  
 1 REFERENCES:

# BLOCK SUMMARY DATA:

BLOCK NUMBER	NUMBER OF PASSES
1	1
2	2
3	2
4	2
5	2
6	3
7	3
8	2
9	1
10	1

TOTAL NUMBER OF BLOCKS PROCESSED WAS 19  
 USING THE STEEPEST DESCENT (MINIMUM CURRENT POOL) BLOCK SELECTION ALGORITHM.  
 TIME FOR THE OPTIMIZATION WAS 0.212 SECONDS.

\*\*\*CODE SYNTHESIS FILTER IS COMPLETE\*\*\*





CARD	BL	SYL	COMMENT
1	0	0	TEST PROGRAM # 2 ;
3	0	1	COMMENT
4	0	1	SAMPLE OF THE STUDENT PROGRAM NUMBER ONE (#1)
5	0	1	THIS PROGRAM READS INPUT CARDS AND THEN TRANSLATES THEM TO
6	0	1	BASIC MATHEMATICAL OPERATIONS. THE FIRST NUMBER ON AN INPUT
7	0	1	CARD INDICATES THE OPERATION DESIRED AND THE NEXT TWO NUMBERS
8	0	1	ARE THE OPERAND (X AND Y). WHEN A CARD IS READ IN WHICH THE
9	0	1	THE MATHEMATICAL OPERATOR CODE IS A 10 THE PROGRAM TERMINATES. ;
10	1	1	BEGIN LOCAL X,Y,Z,A,TEMP,HUNDS,INT,TENS;
11	1	10	READ (X,Y,Z);
12	1	19	WHILE X NEQ 10 DO
13	1	19	BEGIN
14	2	34	IF X EQL 2 THEN A := 100 * (Y + Z)
15	2	37	ELSE
16	2	53	IF X EQL 3 THEN A := 100 * Y * Z
17	2	58	ELSE
18	2	74	IF X EQL 7 THEN A := (100 * Y) / Z
19	2	79	ELSE
20	2	104	IF X EQL 4 THEN A := 100 * (Y - Z);
21	2	111	TEMP := A / 10;
22	2	121	HUNDS := A - TEMP * 10;
23	2	128	INT := A / 100;
24	2	140	TENS := A / 10 - INT * 10;
25	2	149	READ (X, Y, Z);
26	1	152	END;
27	1	152	EOF

CODE FILE COPIED (152 WORDS)

CONSTANT TABLE COPIED (12 WORDS)

2 RECORDS WRITTEN INTO FILE 1

END OF COMPILATION FEBRUARY 12, 1975. CLOCK TIME = 20:53:7.88.

27 CARDS WERE READ.

NO ERRORS WERE DETECTED.



\*\*\*CODE SYNTHESIS FILTER\*\*\*

BLOCK SELECTION METHOD WILL BE LAST-IN-FIRST-OUT

\*\*\*C O M P L E T E T A B L E D U M P\*\*\*  
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 0

CONSTANTS	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
CONSYM	10	2	100	3	7	4								
CONTYPE	INT	INT	INT	INT	INT	INT								
CONVAL	153	154	155	156	157	158								
CONINT	10	2	100	3	7	4								

ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

VALUE STACK (TOP AT 127)

CONT STK	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
BLOCK#	0													
ENTRY	0													

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

\*\*\*\* FORMATTED INTERMEDIATE CODE DUMP \*\*\*\*

LOC	OFF	OP CODE	CNS	ADR	RAW CODE	OPTIMIZATION
SET			/ETC			TYPE
0	0	ENT	C	153	00038099	
1	0	TOGGLE		257	00080101	
2	0	LOC		1	00020001	
3	0	INT		0	00050000	
4	0	STD		0	00290000	
5	0	LOC		2	00020002	
6	0	INT		0	00050000	
7	0	STD		0	00290000	
8	0	LOC		3	00020003	
9	0	INT		0	00050000	
10	0	STD		0	00290000	
11	0	ENT		1	00030001	
12	0	LOC		1	00020001	
13	0	LOD		0	00270000	
14	0	INT		1	00050001	
15	0	NEQ		0	001F0000	
16	0	XIT		152	00040098	
17	0	XIT		19	00040013	
18	0	BSC		0	002F0000	
19	0	ENT		1	00030001	
20	0	LOC		1	00020001	
21	0	LOD		0	00270000	
22	0	INT		2	00050002	
23	0	EQL		0	001E0000	
24	0	XIT		40	00040028	
25	0	XIT		27	00040018	
26	0	BSC		0	002F0000	
27	0	ENT		1	00030001	
28	0	LOC		4	00020004	
29	0	INT		3	00050003	
30	0	LOC		2	00020002	
31	0	LOD		0	00270000	
32	0	LOC		3	00020003	
33	0	LOD		0	00270000	
34	0	ADD		0	00100000	
35	0	MUL		0	00140000	
36	0	STD		0	00280000	
37	0	DEL		0	002A0000	
38	0	XIT		104	00040068	
39	0	BRS		0	002E0000	
40	0	ENT		1	00030001	
41	0	LOC		1	00020001	



42	0	LOU	0	00270000
43	0	INT	4	00050004
44	0	EQL	0	001E0000
45	0	XIT	61	00040030
46	0	XIT	48	00040030
47	0	BSC	0	002F0000
48	0	ENT	1	00030001
49	0	LOC	4	00020004
50	0	INT	3	00050003
51	0	LOC	2	00020002
52	0	LOD	0	00270000
53	0	MUL	0	00140000
54	0	LOC	3	00020003
55	0	LOD	0	00270000
56	0	MUL	0	00140000
57	0	STO	0	00280000
58	0	DEL	0	002A0000
59	0	XIT	103	00040067
60	0	BRS	0	002E0000
61	0	ENT	1	00030001
62	0	LOC	1	00020001
63	0	LOD	0	00270000
64	0	INT	5	00050005
65	0	EQL	0	001E0000
66	0	XIT	82	00040052
67	0	XIT	69	00040045
68	0	BSC	0	002F0000
69	0	ENT	1	00030001
70	0	LOC	4	00020004
71	0	INT	3	00050003
72	0	LOC	2	00020002
73	0	LOD	0	00270000
74	0	MUL	0	00140000
75	0	LOC	3	00020003
76	0	LOD	0	00270000
77	0	DIV	0	00160000
78	0	STO	0	00280000
79	0	DEL	0	002A0000
80	0	XIT	102	00040066
81	0	BRS	0	002E0000
82	0	ENT	1	00030001
83	0	LOC	1	00020001
84	0	LOD	0	00270000
85	0	INT	6	00050006
86	0	EQL	0	001E0000
87	0	XIT	101	00040065
88	0	XIT	90	0004005A
89	0	BSC	0	002F0000
90	0	ENT	1	00030001
91	0	LOC	4	00020004
92	0	INT	3	00050003
93	0	LOC	2	00020002
94	0	LOD	0	00270000
95	0	LOC	3	00020003
96	0	LOD	0	00270000
97	0	SUB	0	00120000
98	0	MUL	0	00140000
99	0	STO	0	00280000
100	0	DEL	0	002A0000
101	0	ENT	1	00030001
102	0	ENT	1	00030001
103	0	ENT	1	00030001
104	0	ENT	1	00030001
105	0	LOC	5	00020005
106	0	LOC	4	00020004
107	0	LOD	0	00270000
108	0	INT	1	00050001
109	0	DIV	0	00160000
110	0	STO	0	00280000
111	0	DEL	0	002A0000
112	0	LOC	6	00020006
113	0	LOC	4	00020004
114	0	LOD	0	00270000
115	0	LOC	5	00020005
116	0	LOD	0	00270000
117	0	INT	1	00050001
118	0	MUL	0	00140000
119	0	SUB	0	00120000
120	0	STO	0	00280000
121	0	DEL	0	002A0000
122	0	LOC	7	00020007
123	0	LOC	4	00020004
124	0	LOD	0	00270000
125	0	INT	3	00050003
126	0	DIV	0	00160000
127	0	STO	0	00280000
128	0	DEL	0	002A0000
129	0	LOC	8	00020008



130	0	LOC	4	00020004
131	0	LOD	0	00270000
132	0	INT	1	00050001
133	0	DIV	0	00160000
134	0	LOC	7	00020007
135	0	LOD	0	00270000
136	0	INT	1	00050001
137	0	MUL	0	00140000
138	0	SUB	0	00120000
139	0	STO	0	00280000
140	0	DEL	0	002A0000
141	0	LOC	1	00020001
142	0	INT	0	00050000
143	0	STD	0	00290000
144	0	LOC	2	00020002
145	0	INT	0	00050000
146	0	STD	0	00290000
147	0	LOC	3	00020003
148	0	INT	0	00050000
149	0	STD	0	00290000
150	0	XIT	11	0004000B
151	0	BRS	0	002E0000
152	0	ENT	1	00030001

\*\*\*\*\*

BASIC BLOCK #1  
 BEGINNING AT 0, ENDING AT 0  
 BLOCK TRAVERSED 0 TIMES  
 0 REFERENCES:

\*\*\*END OF COMPLETE TABLE DUMP\*\*\*

B+2	0	LOC	1	00020001
B+3	0	INT	0	00050000
B+4	0	STD	0	00290000
B+5	0	LOC	2	00020002
B+6	0	INT	0	00050000
B+7	0	STD	0	00290000
B+8	0	LOC	3	00020003
B+9	0	INT	0	00050000
B+10	0	STD	0	00290000
B+11	0	ENT	1	00030001

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 2 IS:  
 A(3,\*,161) A(2,\*,160) A(1,\*,159)

CONTROL AT 11, FROM BLK 1 TO BLK 2 (PASS 1)

C+11	0	ENT	C 157	0003809D
B+12	0	LOC	1	00020001
B+13	0	LOD	0	00270000
B+14	0	INT	1	00050001
B+15	0	NEQ	0	001F0000
B+16	0	XIT	152	00040098
B+17	0	XIT	19	00040013
B+18	0	BSC	0	002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 3 IS:  
 C(159:153,\*,162) A(3,\*,161) A(2,\*,160) A(1,\*,159)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 4 IS:  
 C(159:153,\*,162) A(3,\*,161) A(2,\*,160) A(1,\*,159)

CONTROL AT 19, FROM BLK 2 TO BLK 4 (PASS 1)

C+19	0	ENT	C 167	000380A7
B+20	0	LOC	1	00020001
B+21	0	LOD	0	00270000
B+22	0	INT	2	00050002
B+23	0	EQL	0	001E0000
B+24	0	XIT	40	00040028
B+25	0	XIT	27	0004001B
B+26	0	BSC	0	002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 5 IS:  
 C(159:153,\*,162) C(159:154,\*,163) A(3,\*,161) A(2,\*,160) A(1,\*,159)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 6 IS:  
 C(159:153,\*,162) C(159:154,\*,163) A(3,\*,161) A(2,\*,160) A(1,\*,159)

CONTROL AT 27, FROM BLK 4 TO BLK 6 (PASS 1)

C+27	0	ENT	C 177	000380B1
B+28	0	LOC	4	00020004
B+29	0	INT	3	00050003
B+30	0	LOC	2	00020002
B+31	0	LOD	0	00270000
B+32	0	LOC	3	00020003





B+33	0	LDU	0	00270000
B+34	0	ADD	0	00100000
B+35	0	MUL	0	00140000
B+36	0	STO	0	00280000
B+37	0	DEL	0	002A0000
B+38	0	XIT	104	00040068
B+39	0	BRS	0	002E0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 7 IS:  
 C(159:154,\*,163) C(159:153,\*,162) C(160:161,\*,164) C(155:164,\*,165)  
 A(4,\*,165) A(3,\*,161) A(2,\*,160) A(1,\*,159)

CONTROL AT 104, FROM BLK 6 TO BLK 7 (PASS 1)				
C+104	0	ENT	C 182	00038086
B+105	0	LOC	5	00020005
B+106	0	LOC	4	00020004
B+107	0	LOD	0	00270000
B+108	0	INT	1	00050001
B+109	0	DIV	0	00160000
B+110	0	STO	0	00280000
B+111	0	DEL	0	002A0000
B+112	0	LOC	6	00020006
B+113	0	LOC	4	00020004
B+114	0	LOD	0	00270000
B+115	0	LOC	5	00020005
B+116	0	LOD	0	00270000
B+117	0	INT	1	00050001
B+118	0	MUL	0	00140000
B+119	0	SUB	0	00120000
B+120	0	STO	0	00280000
B+121	0	DEL	0	002A0000
B+122	0	LOC	7	00020007
B+123	0	LOC	4	00020004
B+124	0	LOD	0	00270000
B+125	0	INT	3	00050003
B+126	0	DIV	0	00160000
B+127	0	STO	0	00280000
B+128	0	DEL	0	002A0000
B+129	0	LOC	8	00020008
B+130	0	LOC	4	00020004
B+131	0	LOD	0	00270000
B+132	0	INT	1	00050001
B+133	0	DIV	0	00160000
B+134	0	LOC	7	00020007
B+135	0	LOD	0	00270000
B+136	0	INT	1	00050001
B+137	0	MUL	0	00140000
B+138	0	SUB	0	00120000
B+139	0	STO	0	00280000
B+140	0	DEL	0	002A0000
B+141	0	LOC	1	00020001
B+142	0	INT	0	00050000
B+143	0	STD	0	00290000
B+144	0	LOC	2	00020002
B+145	0	INT	0	00050000
B+146	0	STD	0	00290000
B+147	0	LOC	3	00020003
B+148	0	INT	0	00050000
B+149	0	STD	0	00290000
B+150	0	XIT	11	0004000B
B+151	0	BRS	0	002E0000

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 7 IS:

C(155:164,\*,165) C(160:161,\*,164) C(159:153,\*,162) C(159:154,\*,163)  
 C(159:153,\*,163) C(165:153,\*,166) C(166:153,\*,167) C(155:154,\*,167)  
 C(165:167,\*,168) C(165:155,\*,169) C(169:153,\*,170) C(165:155,\*,170)  
 C(166:170,\*,171) A(3,\*,174) A(2,\*,173) A(1,\*,172) A(8,\*,171) A(7,\*,169)  
 A(6,\*,168) A(5,\*,166) A(4,\*,165)

CURRENT POOL FOR BLOCK# 2 IS:

A(3,\*,161) A(2,\*,160) A(1,\*,159)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

CONTROL AT 40, FROM BLK 4 TO BLK 5 (PASS 1)				
C+40	0	ENT	C 172	000380AC
B+41	0	LOC	1	00020001
B+42	0	LOD	0	00270000
B+43	0	INT	4	00050004
B+44	0	EQL	0	001E0000
B+45	0	XIT	61	0004003D
B+46	0	XIT	48	00040030
B+47	0	BSC	0	002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 2 IS:



INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 8 IS:  
 C(159:154,\*,163) C(159:153,\*,162) C(159:156,\*,178) A(3,\*,161) A(2,\*,160)  
 A(1,\*,159)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 9 IS:  
 C(159:154,\*,163) C(159:153,\*,162) C(159:156,\*,178) A(3,\*,161) A(2,\*,160)  
 A(1,\*,159)

CONTROL AT 48, FROM BLK 5 TO BLK 9 (PASS 1)

C+48	0	ENT	C 192	000380C0
B+49	0	LOC	4	00020004
B+50	0	INT	3	00050003
B+51	0	LOC	2	00020002
B+52	0	LOC	0	00270000
B+53	0	MUL	0	00140000
B+54	0	LUC	3	00020003
B+55	0	LOC	0	00270000
B+56	0	MUL	0	00140000
B+57	0	STO	0	00280000
B+58	0	DEL	0	002A0000
B+59	0	XIT	103	00040067
B+60	0	BRS	0	002EC000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 10 IS:  
 C(159:156,\*,178) C(159:153,\*,162) C(159:154,\*,163) C(155:160,\*,179)  
 C(179:161,\*,180) A(4,\*,180) A(3,\*,161) A(2,\*,160) A(1,\*,159)

CONTROL AT 103, FROM BLK 9 TO BLK 10 (PASS 1)

C+103	0	ENT	C 197	000380C5
B+104	0	ENT	C 182	000380B6

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 10 IS:  
 C(179:161,\*,180) C(155:160,\*,179) C(159:154,\*,163) C(159:153,\*,162)  
 C(159:154,\*,162) C(159:156,\*,178) A(4,\*,180) A(3,\*,161) A(2,\*,160)  
 A(1,\*,159)

CURRENT POOL FOR BLOCK# 7 IS:  
 C(159:154,\*,163) C(159:153,\*,162) C(160:161,\*,164) C(155:164,\*,165)  
 A(4,\*,165) A(3,\*,161) A(2,\*,160) A(1,\*,159)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 7 IS:  
 A(4,\*,181) A(3,\*,161) A(2,\*,160) A(1,\*,159)

CONTROL AT 104, FROM BLK 10 TO BLK 7 (PASS 2)

C+104	0	ENT	C 182	000380B6
B+105	0	LOC	5	00020005
B+106	0	LOC	4	00020004
B+107	0	LOC	0	00270000
B+108	0	INT	1	00050001
B+109	0	DIV	C 988	001683DC
B+110	0	STO	0	00280000
B+111	0	DEL	0	002A0000
B+112	0	LOC	6	00020006
B+113	0	LOC	4	00020004
B+114	0	LOC	0	00270000
B+115	0	LOC	5	00020005
B+116	0	LOC	0	00270000
B+117	0	INT	1	00050001
B+118	0	MUL	C 984	001483D8
B+119	0	SUB	C 980	001283D4
B+120	0	STO	0	00280000
B+121	0	DEL	0	002A0000
B+122	0	LOC	7	00020007
B+123	0	LOC	4	00020004
B+124	0	LOC	0	00270000
B+125	0	INT	3	00050003
B+126	0	DIV	C 976	001683D0
B+127	0	STO	0	00280000
B+128	0	DEL	0	002A0000
B+129	0	LOC	8	00020008
B+130	0	LOC	4	00020004
B+131	0	LOC	0	00270000
B+132	0	INT	1	00050001
B+133	0	DIV	C 972	001683CC
B+134	0	LOC	7	00020007
B+135	0	LOC	0	00270000
B+136	0	INT	1	00050001
B+137	0	MUL	C 968	001483C8
B+138	0	SUB	C 964	001283C4
B+139	0	STO	0	00280000
B+140	0	DEL	0	002A0000

COMM SUBEXP ELIM



B+141	0	LUC	1	00020001
B+142	0	INT	0	00050000
B+143	0	STD	0	00290000
B+144	0	LOC	2	00020002
B+145	0	INT	0	00050000
B+146	0	STD	0	00290000
B+147	0	LOC	3	00020003
B+148	0	INT	0	00050000
B+149	0	STD	0	00290000
B+150	0	XIT	11	0004000E
B+151	0	BRS	0	002E0000

BEFORE THE MEET OPERATION:

OUTPUT PCOL FROM BLOCK# 7 IS:

C(181:153,\*,182) C(182:153,\*,183) C(181:183,\*,184) C(181:155,\*,185)  
 C(185:153,\*,186) C(181:155,\*,188) C(182:186,\*,187) A(3,\*,190) A(2,\*,189)  
 A(1,\*,188) A(8,\*,187) A(7,\*,185) A(6,\*,184) A(5,\*,182) A(4,\*,181)

CURRENT POOL FOR BLOCK# 2 IS:

A(3,\*,175) A(2,\*,176) A(1,\*,177)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

CONTROL AT 61, FROM BLK 5 TO BLK 8 (PASS 1)

C+61	0	ENT	C 187	000380BB
B+62	0	LOC	1	00020001
B+63	0	LOC	0	00270000
B+64	0	INT	5	00050005
B+65	0	EQL	0	001E0000
B+66	0	XIT	82	00040052
B+67	0	XIT	69	00040045
B+68	0	BSC	0	002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 11 IS:

C(159:156,\*,178) C(159:153,\*,162) C(159:154,\*,163) C(159:157,\*,194)  
 A(3,\*,161) A(2,\*,160) A(1,\*,159)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 12 IS:

C(159:156,\*,178) C(159:153,\*,162) C(159:154,\*,163) C(159:157,\*,194)  
 A(3,\*,161) A(2,\*,160) A(1,\*,159)

CONTROL AT 69, FROM BLK 8 TO BLK 12 (PASS 1)

C+69	0	ENT	C 207	000380CF
B+70	0	LOC	4	00020004
B+71	0	INT	3	00050003
B+72	0	LOC	2	00020002
B+73	0	LOC	0	00270000
B+74	0	MUL	0	00140000
B+75	0	LOC	3	00020003
B+76	0	LOC	0	00270000
B+77	0	DIV	0	00160000
B+78	0	STD	0	00280000
B+79	0	DEL	0	002A0000
B+80	0	XIT	102	00040066
B+81	0	BRS	0	002E0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 13 IS:

C(159:157,\*,194) C(159:154,\*,163) C(159:153,\*,162) C(159:156,\*,178)  
 C(155:160,\*,195) C(195:161,\*,196) A(4,\*,196) A(3,\*,161) A(2,\*,160)  
 A(1,\*,159)

CONTROL AT 102, FROM BLK 12 TO BLK 13 (PASS 1)

C+102	0	ENT	C 212	000380D4
B+103	0	ENT	C 197	000380C5

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 13 IS:

C(195:161,\*,196) C(155:160,\*,195) C(159:156,\*,178) C(159:153,\*,162)  
 C(159:154,\*,163) C(159:153,\*,163) C(159:157,\*,194) A(4,\*,196) A(3,\*,161)  
 A(2,\*,160) A(1,\*,159)

CURRENT POOL FOR BLOCK# 10 IS:

C(159:156,\*,178) C(159:153,\*,162) C(159:154,\*,163) C(155:160,\*,179)  
 C(179:161,\*,180) A(4,\*,180) A(3,\*,161) A(2,\*,160) A(1,\*,159)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 10 IS:

A(4,\*,197) A(3,\*,161) A(2,\*,160) A(1,\*,159)

CONTROL AT 103, FROM BLK 13 TO BLK 10 (PASS 2)





```
C+103 0 ENT C197 000380C5
B+104 0 ENT C182 000380B6
```

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 10 IS:

A(4,\*,197) A(3,\*,161) A(2,\*,160) A(1,\*,159)

CURRENT POOL FOR BLOCK# 7 IS:

A(4,\*,181) A(3,\*,161) A(2,\*,160) A(1,\*,159)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

```
CONTROL AT 82, FROM BLK 8 TO BLK 11 (PASS 1)
C+82 0 ENT C202 000380CA
B+83 0 LOC 1 00020001
B+84 0 LOD 0 00270000
B+85 0 INT 6 00050006
B+86 0 EQL 0 001E0000
B+87 0 XIT 101 00040065
B+88 0 XIT 90 0004005A
B+89 0 BSC 0 002F0000
```

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 14 IS:

C(159:157,\*,194) C(159:154,\*,163) C(159:153,\*,162) C(159:156,\*,178)  
C(159:158,\*,199) A(3,\*,161) A(2,\*,160) A(1,\*,159)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 15 IS:

C(159:157,\*,194) C(159:154,\*,163) C(159:153,\*,162) C(159:156,\*,178)  
C(159:158,\*,199) A(3,\*,161) A(2,\*,160) A(1,\*,159)

```
CONTROL AT 90, FROM BLK 11 TO BLK 15 (PASS 1)
C+90 0 ENT C222 000380DE
B+91 0 LOC 4 00020004
B+92 0 INT 3 00050003
B+93 0 LOC 2 00020002
B+94 0 LOD 0 00270000
B+95 0 LOC 3 00020003
B+96 0 LOD 0 00270000
B+97 0 SUB 0 00120000
B+98 0 MUL 0 00140000
B+99 0 STO 0 00280000
B+100 0 DEL 0 002A0000
B+101 0 ENT C217 000380D9
```

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 15 IS:

C(159:158,\*,199) C(159:156,\*,178) C(159:153,\*,162) C(159:154,\*,163)  
C(159:153,\*,163) C(159:157,\*,194) C(160:161,\*,200) C(155:200,\*,201)  
A(4,\*,201) A(3,\*,161) A(2,\*,160) A(1,\*,159)

CURRENT POOL FOR BLOCK# 14 IS:

C(159:157,\*,194) C(159:154,\*,163) C(159:153,\*,162) C(159:156,\*,178)  
C(159:158,\*,199) A(3,\*,161) A(2,\*,160) A(1,\*,159)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 14 IS:

A(3,\*,161) A(2,\*,160) A(1,\*,159)

```
CONTROL AT 101, FROM BLK 15 TO BLK 14 (PASS 1)
C+101 0 ENT C217 000380D9
B+102 0 ENT C212 000380D4
```

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 14 IS:

A(3,\*,161) A(2,\*,160) A(1,\*,159)

CURRENT POOL FOR BLOCK# 13 IS:

C(159:157,\*,194) C(159:154,\*,163) C(159:153,\*,162) C(159:156,\*,178)  
C(155:160,\*,195) C(195:161,\*,196) A(4,\*,196) A(3,\*,161) A(2,\*,160)  
A(1,\*,159)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 13 IS:

A(3,\*,161) A(2,\*,160) A(1,\*,159)

CONTROL AT 102, FROM BLK 14 TO BLK 13 (PASS 2)





C+102	0	ENT	C 212	000380D4
B+103	0	ENT	C 197	000380C5

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 13 IS:  
A(3,\*,161) A(2,\*,160) A(1,\*,159)

CURRENT POOL FOR BLOCK# 10 IS:  
A(4,\*,197) A(3,\*,161) A(2,\*,160) A(1,\*,159)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 10 IS:  
A(3,\*,161) A(2,\*,160) A(1,\*,159)

CONTROL AT 103, FROM BLK 13 TO BLK 10 (PASS 3)

C+103	0	ENT	C 197	000380C5
B+104	0	ENT	C 182	000380B6

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 10 IS:  
A(3,\*,161) A(2,\*,160) A(1,\*,159)

CURRENT POOL FOR BLOCK# 7 IS:  
A(4,\*,198) A(3,\*,161) A(2,\*,160) A(1,\*,159)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 7 IS:  
A(3,\*,161) A(2,\*,160) A(1,\*,159)

CONTROL AT 104, FROM BLK 10 TO BLK 7 (PASS 3)

C+104	0	ENT	C 182	000380B6
B+105	0	LOC	5	00020005
B+106	0	LOC	4	00020004
B+107	0	LOC	0	00270000
R+108	0	INT	1	00050001
B+109	0	DIV	C 988	001683DC
B+110	0	STO	0	00280000
B+111	0	DEL	0	002A0000
B+112	0	LOC	6	00020006
B+113	0	LOC	4	00020004
B+114	0	LOC	0	00270000
B+115	0	LOC	5	00020005
B+116	0	LOC	0	00270000
B+117	0	INT	1	00050001
B+118	0	MUL	C 984	001483D8
B+119	0	SUB	C 980	001283D4
B+120	0	STO	0	00280000
B+121	0	DEL	0	002A0000
B+122	0	LOC	7	00020007
B+123	0	LOC	4	00020004
B+124	0	LOC	0	00270000
B+125	0	INT	3	00050003
B+126	0	DIV	C 976	001683D0
B+127	0	STO	0	00280000
B+128	0	DEL	0	002A0000
B+129	0	LOC	8	00020008
B+130	0	LOC	4	00020004
B+131	0	LOC	0	00270000
B+132	0	INT	1	00050001
B+133	0	DIV	C 972	001683CC
B+134	0	LOC	7	00020007
B+135	0	LOC	0	00270000
B+136	0	INT	1	00050001
B+137	0	MUL	C 968	001483C8
B+138	0	SUB	C 964	001283C4
B+139	0	STO	0	00280000
B+140	0	DEL	0	002A0000
B+141	0	LOC	1	00020001
B+142	0	INT	0	00050000
B+143	0	STO	0	00280000
B+144	0	LOC	2	00020002
B+145	0	INT	0	00050000
B+146	0	STO	0	00290000
B+147	0	LOC	3	00020003
B+148	0	INT	0	00050000
B+149	0	STO	0	00290000
B+150	0	XIT	11	00040008
B+151	0	BRS	0	002E0000

COMM SUBEXP ELIM

BEFORE THE MEET OPERATION:



OUTPUT POOL FROM BLOCK# 7 IS:  
 C(202:153,\*,203) C(203:153,\*,204) C(202:204,\*,205) C(202:155,\*,206)  
 C(206:153,\*,207) C(202:155,\*,207) C(203:207,\*,208) A(3,\*,211) A(2,\*,210)  
 A(1,\*,209) A(8,\*,208) A(7,\*,206) A(6,\*,205) A(4,\*,202) A(5,\*,203)

CURRENT POOL FOR BLOCK# 2 IS:  
 A(3,\*,191) A(2,\*,192) A(1,\*,193)

AFTER PERFORMING THE MEET OPERATION:  
 NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

CONTROL AT 152, FROM BLK 2 TO BLK 3 (PASS 1)  
 C+152 0 IENT IC162 000380A2

# FINAL OPTIMIZATION RESULTS

OPTIMIZATION AT 133  
 (153,202|109,203) COMM SUBEXP ELIM

**** FORMATTED INTERMEDIATE CODE DUMP ****					
LOC	OFF	OP CODE	CNS ADR	RAW CODE	OPTIMIZATION TYPE
SET			/ETC		
0	0	ENT	153	00030099	
1	0	TOGGLE	257	00080101	
2	0	LOC	1	00020001	
3	0	INT	0	00050000	
4	0	STD	0	00290000	
5	0	LOC	2	00020002	
6	0	INT	0	00050000	
7	0	STD	0	00290000	
8	0	LOC	3	00020003	
9	0	INT	0	00050000	
10	0	STD	0	00290000	
11	0	ENT	157	0003009D	
12	0	LOC	1	00020001	
13	0	LOD	0	00270000	
14	0	INT	1	00050001	
15	0	NEQ	1024	001F0400	
16	0	XIT	152	00040098	
17	0	XIT	19	00040013	
18	0	BSC	0	002F0000	
19	0	ENT	167	000300A7	
20	0	LOC	1	00020001	
21	0	LOD	0	00270000	
22	0	INT	2	00050002	
23	0	EQL	1016	001E03F8	
24	0	XIT	40	00040028	
25	0	XIT	27	0004001B	
26	0	BSC	0	002F0000	
27	0	ENT	177	000300B1	
28	0	LOC	4	00020004	
29	0	INT	3	00050003	
30	0	LOC	2	00020002	
31	0	LOD	0	00270000	
32	0	LOC	3	00020003	
33	0	LOD	0	00270000	
34	0	ADD	1004	001003EC	
35	0	MUL	1000	001403E8	
36	0	STO	0	002E0000	
37	0	DEL	0	002A0000	
38	0	XIT	104	00040068	
39	0	BRS	0	002E0000	
40	0	ENT	172	000300AC	
41	0	LOC	1	00020001	
42	0	LOD	0	00270000	
43	0	INT	4	00050004	
44	0	EQL	960	001E03C0	
45	0	XIT	61	0004003D	
46	0	XIT	48	00040030	
47	0	BSC	0	002F0000	
48	0	ENT	192	000300C0	
49	0	LOC	4	00020004	
50	0	INT	3	00050003	
51	0	LOC	2	00020002	
52	0	LOD	0	00270000	
53	0	MUL	944	001403B0	
54	0	LOC	3	00020003	
55	0	LOD	0	00270000	
56	0	MUL	940	001403AC	
57	0	STO	0	002E0000	



58	0	DEL	0	002A0000
59	0	XIT	103	00040067
60	0	BRS	0	002E0000
61	0	ENT	187	00030088
62	0	LOC	1	00020001
63	0	LOD	0	00270000
64	0	INT	5	00050005
65	0	EQL	926	001E039E
66	0	XIT	82	00040052
67	0	XIT	69	00040045
68	0	BSC	0	002F0000
69	0	ENT	207	000300CF
70	0	LOC	4	00020004
71	0	INT	3	00050003
72	0	LOC	2	00020002
73	0	LOD	0	00270000
74	0	MUL	906	0014038A
75	0	LOC	3	00020003
76	0	LOD	0	00270000
77	0	DIV	902	00160386
78	0	STO	0	00280000
79	0	DEL	0	002A0000
80	0	XIT	102	00040066
81	0	BRS	0	002E0000
82	0	ENT	202	000300CA
83	0	LOC	1	00020001
84	0	LOD	0	00270000
85	0	INT	6	00050006
86	0	EQL	886	001E0376
87	0	XIT	101	00040065
88	0	XIT	90	0004005A
89	0	BSC	0	002F0000
90	0	ENT	222	000300DE
91	0	LOC	4	00020004
92	0	INT	3	00050003
93	0	LCC	2	00020002
94	0	LOD	0	00270000
95	0	LOC	3	00020003
96	0	LOD	0	00270000
97	0	SUB	862	0012035E
98	0	MUL	858	0014035A
99	0	STO	0	00280000
100	0	DEL	0	002A0000
101	0	ENT	217	000300D9
102	0	ENT	212	000300D4
103	0	ENT	197	000300C5
104	0	ENT	182	000300B6
105	0	LOC	5	00020005
106	0	LOC	4	00020004
107	0	LOD	0	00270000
108	0	INT	1	00050001
109	0	DIV	988	001603DC
110	0	STO	0	00280000
111	0	DEL	0	002A0000
112	0	LOC	6	00020006
113	0	LOC	4	00020004
114	0	LOD	0	00270000
115	0	LOC	5	00020005
116	0	LOD	0	00270000
117	0	INT	1	00050001
118	0	MUL	984	00140308
119	0	SUB	980	00120304
120	0	STO	0	00280000
121	0	DEL	0	002A0000
122	0	LOC	7	00020007
123	0	LOC	4	00020004
124	0	LOD	0	00270000
125	0	INT	3	00050003
126	0	DIV	976	00160300
127	0	STO	0	00280000
128	0	DEL	0	002A0000
129	0	LOC	8	00020008
130	0	LOC	4	00020004
131	0	LOD	0	00270000
132	0	INT	1	00050001
133	0	DIV	972	001683CC
134	0	LOC	7	00020007
135	0	LOD	0	00270000
136	0	INT	1	00050001
137	0	MUL	968	001403C8
138	0	SUB	964	001203C4
139	0	STO	0	00280000
140	0	DEL	0	002A0000
141	0	LOC	1	00020001
142	0	INT	0	00050000
143	0	STO	0	00290000
144	0	LOC	2	00020002
145	0	INT	0	00050000

COMM SUBEXP ELIM





146	U	STD	0	00290000
147	0	LOC	3	00020003
148	0	INT	0	00050000
149	0	STD	0	00290000
150	0	XIT	11	00040008
151	0	BRS	0	002E0000
152	0	ENT	162	000300A2

\*\*\*\*\*

FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

FINAL POCL FOR BLOCK# 1 IS:  
(NULL)

FINAL POCL FOR BLOCK# 2 IS:  
A(3,\*,212) A(2,\*,213) A(1,\*,214)

FINAL POCL FOR BLOCK# 3 IS:  
C(159:153,\*,162) A(3,\*,161) A(2,\*,160) A(1,\*,159)

FINAL POCL FOR BLOCK# 4 IS:  
C(159:153,\*,162) A(3,\*,161) A(2,\*,160) A(1,\*,159)

FINAL POCL FOR BLOCK# 5 IS:  
C(159:153,\*,162) C(159:154,\*,163) A(3,\*,161) A(2,\*,160) A(1,\*,159)

FINAL POCL FOR BLOCK# 6 IS:  
C(159:153,\*,162) C(159:154,\*,163) A(3,\*,161) A(2,\*,160) A(1,\*,159)

FINAL POCL FOR BLOCK# 7 IS:  
A(3,\*,161) A(2,\*,160) A(1,\*,159)

FINAL POCL FOR BLOCK# 8 IS:  
C(159:154,\*,163) C(159:153,\*,162) C(159:156,\*,178) A(3,\*,161) A(2,\*,160)  
A(1,\*,159)

FINAL POCL FOR BLOCK# 9 IS:  
C(159:154,\*,163) C(159:153,\*,162) C(159:156,\*,178) A(3,\*,161) A(2,\*,160)  
A(1,\*,159)

FINAL POCL FOR BLOCK# 10 IS:  
A(3,\*,161) A(2,\*,160) A(1,\*,159)

FINAL POCL FOR BLOCK# 11 IS:  
C(159:156,\*,178) C(159:153,\*,162) C(159:154,\*,163) C(159:157,\*,194)  
A(3,\*,161) A(2,\*,160) A(1,\*,159)

FINAL POCL FOR BLOCK# 12 IS:  
C(159:156,\*,178) C(159:153,\*,162) C(159:154,\*,163) C(159:157,\*,194)  
A(3,\*,161) A(2,\*,160) A(1,\*,159)

FINAL POCL FOR BLOCK# 13 IS:  
A(3,\*,161) A(2,\*,160) A(1,\*,159)

FINAL POCL FOR BLOCK# 14 IS:  
A(3,\*,161) A(2,\*,160) A(1,\*,159)





FINAL POCL FOR BLOCK# 15 IS:

C(159:157,\*,194) C(159:154,\*,163) C(159:153,\*,162) C(159:156,\*,178)  
C(159:158,\*,199) A(3,\*,161) A(2,\*,160) A(1,\*,159)

BASIC BLOCK #1  
BEGINNING AT 0, ENDING AT 11  
BLOCK TRAVERSED 1 TIMES  
0 REFERENCES:  
1  
BASIC BLOCK #2  
BEGINNING AT 11, ENDING AT 18  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
2  
BASIC BLOCK #3  
BEGINNING AT 152, ENDING AT 152  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
2  
BASIC BLOCK #4  
BEGINNING AT 19, ENDING AT 26  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
2  
BASIC BLOCK #5  
BEGINNING AT 40, ENDING AT 47  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
4  
BASIC BLOCK #6  
BEGINNING AT 27, ENDING AT 39  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
4  
BASIC BLOCK #7  
BEGINNING AT 104, ENDING AT 151  
BLOCK TRAVERSED 3 TIMES  
1 REFERENCES:  
6  
BASIC BLOCK #8  
BEGINNING AT 61, ENDING AT 68  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
5  
BASIC BLOCK #9  
BEGINNING AT 48, ENDING AT 60  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
5  
BASIC BLOCK #10  
BEGINNING AT 103, ENDING AT 104  
BLOCK TRAVERSED 3 TIMES  
1 REFERENCES:  
9  
BASIC BLOCK #11  
BEGINNING AT 82, ENDING AT 89  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
8  
BASIC BLOCK #12  
BEGINNING AT 69, ENDING AT 81  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
8  
BASIC BLOCK #13  
BEGINNING AT 102, ENDING AT 103  
BLOCK TRAVERSED 2 TIMES  
1 REFERENCES:  
12  
BASIC BLOCK #14  
BEGINNING AT 101, ENDING AT 102  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
15  
BASIC BLOCK #15  
BEGINNING AT 90, ENDING AT 101  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
11

BLOCK SUMMARY DATA:

BLOCK NUMBER	NUMBER OF PASSES
1	1



2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15

1  
1  
1  
1  
1  
3  
1  
1  
1  
3  
1  
1  
2  
1  
1

TOTAL NUMBER OF BLOCKS PROCESSED WAS 20  
USING THE LAST-IN-FIRST-OUT BLOCK SELECTION ALGORITHM.  
TIME FOR THE OPTIMIZATION WAS 1.411 SECONDS.

\*\*\*CODE SYNTHESIS FILTER IS COMPLETE\*\*\*



CARD	BL	SYL	COMMENT
1	0	0	TEST PROGRAM # 3 ;
3	0	1	\$EXECUTE BEGIN LOCAL A,B,C,D,E,F,G,H,I;
4	1	1	B := 5;
5	1	5	C := 10;
6	1	9	A := B * C;
7	1	17	IF A GTR B THEN
8	1	22	BEGIN
9	1	26	C := 30;
10	2	30	E := C;
11	2	35	G := B * E;
12	2	43	END ELSE
13	1	43	BEGIN
14	1	46	B := B + 2;
15	2	53	D := B;
16	2	58	F := D * C;
17	2	66	END;
18	1	67	H := C * B;
19	1	75	I := B * C;
20	1	83	WHILE H GTR C DO
21	1	93	BEGIN
22	1	93	C := C + 1;
23	2	100	F := 1 + 2;
24	2	106	G := G + 0;
25	2	113	A := A * 1;
26	2	120	B := B * 0;
27	2	127	END;
28	1	130	END
29	1	130	EOF

CODE FILE COPIED (130 WORDS)

CONSTANT TABLE COPIED (12 WORDS)

2 RECORDS WRITTEN INTO FILE 1

END OF COMPILATION FEBRUARY 12, 1975. CLOCK TIME = 20:29:35.48.

29 CARDS WERE READ.

NO ERRORS WERE DETECTED.



\*\*\*CODE SYNTHESIS FILTER\*\*\*

BLOCK SELECTION METHOD WILL BE FIRST-IN-FIRST-OUT

\*\*\*C O M P L E T E T A B L E D U M P\*\*\*  
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 0

CONSTANTS	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
CONSYM	5	10	30	2	1	0								
CONTYPE	INT	INT	INT	INT	INT	INT								
CONVAL	131	132	133	134	135	136								
CONINT	5	10	30	2	1	0								

ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

VALUE STACK (TOP AT 127)

CONT STK	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
BLOCK#	0													
ENTRY	0													

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

\*\*\*\* FORMATTED INTERMEDIATE CODE DUMP \*\*\*\*

LOC	OFF	OP CODE	CNS ADDR	RAW CODE	OPTIMIZATION TYPE
0	0	ENT	131	00038083	
1	0	TOGGLE	257	000R0101	
2	0	LOC	2	00020002	
3	0	INT	1	00050001	
4	0	STO	0	00280000	
5	0	DEL	0	002A0000	
6	0	LOC	3	00020003	
7	0	INT	2	00050002	
8	0	STO	0	00280000	
9	0	DEL	0	002A0000	
10	0	LOC	1	00020001	
11	0	LOC	2	00020002	
12	0	LOD	0	00270000	
13	0	LOC	3	00020003	
14	0	LOD	0	00270000	
15	0	MUL	0	00140000	
16	0	STO	0	00280000	
17	0	DEL	0	002A0000	
18	0	LOC	1	00020001	
19	0	LOD	0	00270000	
20	0	LOC	2	00020002	
21	0	LOD	0	00270000	
22	0	GTR	0	00210000	
23	0	XIT	46	0004002E	
24	0	XIT	26	0004001A	
25	0	BSC	0	002F0000	
26	0	ENT	1	00030001	
27	0	LOC	3	00020003	
28	0	INT	3	00050003	
29	0	STO	0	00280000	
30	0	DEL	0	002A0000	
31	0	LOC	5	00020005	
32	0	LOC	3	00020003	
33	0	LOD	0	00270000	
34	0	STO	0	00280000	
35	0	DEL	0	002A0000	
36	0	LOC	7	00020007	
37	0	LOC	2	00020002	
38	0	LOD	0	00270000	
39	0	LOC	5	00020005	
40	0	LOD	0	00270000	
41	0	MUL	0	00140000	





42	U	STU	U	00280000
43	0	DEL	0	002A0000
44	0	XIT	67	00040043
45	0	BRS	0	002E0000
46	0	ENT	1	00030001
47	0	LOC	2	00020002
48	0	LOC	2	00020002
49	0	LOD	0	00270000
50	0	INT	4	00050004
51	0	ADD	0	00100000
52	0	STO	0	00280000
53	0	DEL	0	002A0000
54	0	LOC	4	00020004
55	0	LOC	2	00020002
56	0	LOD	0	00270000
57	0	STO	0	00280000
58	0	DEL	0	002A0000
59	0	LOC	6	00020006
60	0	LOC	4	00020004
61	0	LOD	0	00270000
62	0	LOC	3	00020003
63	0	LOD	0	00270000
64	0	MUL	0	00140000
65	0	STO	0	00280000
66	0	DEL	0	002A0000
67	0	ENT	1	00030001
68	0	LOC	8	00020008
69	0	LOC	3	00020003
70	0	LOD	0	00270000
71	0	LOC	2	00020002
72	0	LOD	0	00270000
73	0	MUL	0	00140000
74	0	STO	0	00280000
75	0	DEL	0	002A0000
76	0	LOC	9	00020009
77	0	LOC	2	00020002
78	0	LOD	0	00270000
79	0	LOC	3	00020003
80	0	LOD	0	00270000
81	0	MUL	0	00140000
82	0	STO	0	00280000
83	0	DEL	0	002A0000
84	0	ENT	1	00030001
85	0	LOC	8	00020008
86	0	LOD	0	00270000
87	0	LOC	3	00020003
88	0	LOD	0	00270000
89	0	GTR	0	00210000
90	0	XIT	130	00040082
91	0	XIT	93	00040050
92	0	BSC	0	002F0000
93	0	ENT	1	00030001
94	0	LOC	3	00020003
95	0	LOC	3	00020003
96	0	LOD	0	00270000
97	0	INT	5	00050005
98	0	ADD	0	00100000
99	0	STO	0	00280000
100	0	DEL	0	002A0000
101	0	LOC	6	00020006
102	0	INT	5	00050005
103	0	INT	4	00050004
104	0	ADD	0	00100000
105	0	STO	0	00280000
106	0	DEL	0	002A0000
107	0	LOC	7	00020007
108	0	LOC	7	00020007
109	0	LOD	0	00270000
110	0	INT	6	00050006
111	0	ADD	0	00100000
112	0	STO	0	00280000
113	0	DEL	0	002A0000
114	0	LOC	1	00020001
115	0	LOC	1	00020001
116	0	LOD	0	00270000
117	0	INT	5	00050005
118	0	MUL	0	00140000
119	0	STO	0	00280000
120	0	DEL	0	002A0000
121	0	LOC	2	00020002
122	0	LOC	2	00020002
123	0	LOD	0	00270000
124	0	INT	6	00050006
125	0	MUL	0	00140000
126	0	STO	0	00280000
127	0	DEL	0	002A0000
128	0	XIT	84	00040054
129	0	BRS	0	002E0000



```

130 0 1EN1 1 11 1 00030001 1
*****

```

BASIC BLOCK #1  
 BEGINNING AT 0, ENDING AT 0  
 BLOCK TRAVERSED 0 TIMES  
 0 REFERENCES:

\*\*\*END OF COMPLETE TABLE DUMP\*\*\*

B+2	0	LOC	2	00020002
B+3	0	INT	1	00050001
B+4	0	STO	0	00280000
B+5	0	DEL	0	002A0000
B+6	0	LOC	3	00020003
B+7	0	INT	2	00050002
B+8	0	STO	0	00280000
B+9	0	DEL	0	002A0000
B+10	0	LOC	1	00020001
B+11	0	LOC	2	00020002
B+12	0	LOD	0	00270000
B+13	0	LOC	3	00020003
B+14	0	LOD	0	00270000
B+15	0	MUL	0	00140000
B+16	0	STO	0	00280000
B+17	0	DEL	0	002A0000
B+18	0	LOC	1	00020001
B+19	0	LOD	0	00270000
B+20	0	LOC	2	00020002
B+21	0	LOD	0	00270000
B+22	0	GTR	0	00210000
B+23	0	XIT	46	0004002E
B+24	0	XIT	26	0004001A
B+25	0	BSC	0	002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 2 IS:  
 C(131:132,\*,138) C(138:131,\*,140) A(1,138,138) A(3,132,132) A(2,131,131)

CONTROL AT 26, FROM BLK 1 TO BLK 2 (PASS 1)

C+26	0	ENT	C 135	00038087
B+27	0	LOC	3	00020003
B+28	0	INT	3	00050003
B+29	0	STO	0	00280000
B+30	0	DEL	0	002A0000
B+31	0	LOC	5	00020005
B+32	0	LOC	3	00020003
B+33	0	LOD	0	00270000
B+34	0	STO	0	00280000
B+35	0	DEL	0	002A0000
B+36	0	LOC	7	00020007
B+37	0	LOC	2	00020002
B+38	0	LOD	0	00270000
B+39	0	LOC	5	00020005
B+40	0	LOD	0	00270000
B+41	0	MUL	0	00140000
B+42	0	STO	0	00280000
B+43	0	DEL	0	002A0000
B+44	0	XIT	67	00040043
B+45	0	BRS	0	002E0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 3 IS:  
 C(138:131,\*,140) C(131:132,\*,138) C(131:133,\*,142) A(7,142,142)  
 A(5,133,133) A(3,133,133) A(1,138,138) A(2,131,131)

CONTROL AT 67, FROM BLK 2 TO BLK 3 (PASS 1)

C+67	0	ENT	C 140	0003808C
B+68	0	LOC	8	00020008
B+69	0	LOC	3	00020003
B+70	0	LOD	0	00270000
B+71	0	LOC	2	00020002
B+72	0	LOD	0	00270000
B+73	0	MUL	0	00140000
B+74	0	STO	0	00280000
B+75	0	DEL	0	002A0000
B+76	0	LOC	9	00020009
B+77	0	LOC	2	00020002
B+78	0	LOD	0	00270000
B+79	0	LOC	3	00020003
B+80	0	LOD	0	00270000
B+81	0	MUL	0	00140000
B+82	0	STO	0	00280000
B+83	0	DEL	0	002A0000
B+84	0	ENT	1	00030001

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 4 IS:  
 C(131:133,\*,142) C(131:132,\*,138) C(138:131,\*,140) A(9,142,142)  
 A(8,142,142) A(7,142,142) A(5,133,133) A(3,133,133) A(1,138,138)



A(2,131,131)

CONTROL AT 84, FROM BLK 3 TO BLK 4 (PASS 1)			
C+84	0	ENT	C 145
B+85	0	LOC	8
B+86	0	LOD	0
B+87	0	LOC	3
B+88	0	LOD	0
B+89	0	GTR	0
B+90	0	XIT	130
B+91	0	XIT	93
B+92	0	BSC	0

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 5 IS:  
 C(138:131,\*,140) C(131:132,\*,138) C(131:133,\*,142) C(142:133,\*,140)  
 A(9,142,142) A(8,142,142) A(7,142,142) A(5,133,133) A(3,133,133)  
 A(1,138,138) A(2,131,131)

CONTROL AT 93, FROM BLK 4 TO BLK 5 (PASS 1)			
C+93	0	ENT	C 150
B+94	0	LOC	3
B+95	0	LOC	3
B+96	0	LOD	0
B+97	0	INT	5
B+98	0	ADD	0
B+99	0	STO	0
B+100	0	DEL	0
B+101	0	LOC	6
B+102	0	INT	5
B+103	0	INT	4
B+104	0	ADD	0
B+105	0	STO	0
B+106	0	DEL	0
B+107	0	LOC	7
B+108	0	LOC	7
B+109	0	LOD	0
B+110	0	INT	6
B+111	0	ADD	0
B+112	0	STO	0
B+113	0	DEL	0
B+114	0	LOC	1
B+115	0	LOC	1
B+116	0	LOD	0
B+117	0	INT	5
B+118	0	MUL	0
B+119	0	STO	0
B+120	0	DEL	0
B+121	0	LOC	2
B+122	0	LOC	2
B+123	0	LOD	0
B+124	0	INT	6
B+125	0	MUL	0
B+126	0	STO	0
B+127	0	DEL	0
B+128	0	XIT	84
B+129	0	BRS	0

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 5 IS:  
 C(142:133,\*,140) C(131:133,\*,142) C(131:132,\*,138) C(138:131,\*,140)  
 C(133:135,\*,145) C(131:133,\*,145) C(135:134,\*,147) C(142:136,\*,142)  
 C(136:135,\*,138) C(131:136,\*,136) A(2,136,136) A(1,138,138) A(7,142,142)  
 A(6,147,147) A(3,145,145) A(9,142,142) A(8,142,142) A(5,133,133)

CURRENT POOL FOR BLOCK# 4 IS:  
 C(131:133,\*,142) C(131:132,\*,138) C(138:131,\*,140) A(9,142,142)  
 A(8,142,142) A(7,142,142) A(5,133,133) A(3,133,133) A(1,138,138)  
 A(2,131,131)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 4 IS:  
 A(9,142,142) A(8,142,142) A(7,142,142) A(5,133,133) A(3,\*,151)  
 A(1,138,138) A(2,\*,152)

CONTROL AT 84, FROM BLK 5 TO BLK 4 (PASS 2)			
C+84	0	ENT	C 145
B+85	0	LOC	8
B+86	0	LOD	0
B+87	0	LOC	3
B+88	0	LOD	0
B+89	0	GTR	C 988
B+90	0	XIT	130
B+91	0	XIT	93
B+92	0	BSC	0

CONSTANT. PROPAGATION





INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 6 IS:  
 C(142:151,\*,153) A(9,142,142) A(8,142,142) A(7,142,142) A(5,133,133)  
 A(3,\*,151) A(1,138,138) A(2,\*,152)

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 4 IS:  
 C(142:151,\*,153) A(9,142,142) A(8,142,142) A(7,142,142) A(5,133,133)  
 A(3,\*,151) A(1,138,138) A(2,\*,152)

CURRENT POOL FOR BLOCK# 5 IS:  
 C(138:131,\*,140) C(131:132,\*,138) C(131:133,\*,142) C(142:133,\*,140)  
 A(9,142,142) A(8,142,142) A(7,142,142) A(5,133,133) A(3,133,133)  
 A(1,138,138) A(2,131,131)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 5 IS:  
 A(9,142,142) A(8,142,142) A(7,142,142) A(5,133,133) A(3,\*,154)  
 A(1,138,138) A(2,\*,155) C(142:154,\*,156)

CONTROL AT 130, FROM BLK 4 TO BLK 6 (PASS 1)			
C+130	0	ENT	C(155 00038098
CONTROL AT 93, FROM BLK 4 TO BLK 5 (PASS 2)			
C+93	0	ENT	C 150 00038096
B+94	0	LOC	3 00020003
B+95	0	LOC	3 00020003
B+96	0	LOD	0 00270000
B+97	0	INT	5 00050005
B+98	0	ADD	C 976 001083D0
B+99	0	STO	0 00280000
B+100	0	DEL	0 002A0000
B+101	0	LOC	6 00020006
B+102	0	INT	5 00050005
B+103	0	INT	4 00050004
B+104	0	ADD	C 972 001083CC
B+105	0	STO	0 00280000
B+106	0	DEL	0 002A0000
B+107	0	LOC	7 00020007
B+108	0	LOC	7 00020007
B+109	0	LOD	0 00270000
B+110	0	INT	6 00050006
B+111	0	ADD	C 968 001083C8
B+112	0	STO	0 00280000
B+113	0	DEL	0 002A0000
B+114	0	LOC	1 00020001
B+115	0	LOC	1 00020001
B+116	0	LOD	0 00270000
B+117	0	INT	5 00050005
B+118	0	MUL	C 964 001483C4
B+119	0	STO	0 00280000
B+120	0	DEL	0 002A0000
B+121	0	LOC	2 00020002
B+122	0	LOC	2 00020002
B+123	0	LOD	0 00270000
B+124	0	INT	6 00050006
B+125	0	MUL	C 960 001483C0
B+126	0	STO	0 00280000
B+127	0	DEL	0 002A0000
B+128	0	XIT	84 00040054
B+129	0	BRS	0 002E0000

CONSTANT PROPAGATION

CONSTANT PROPAGATION

CONSTANT PROPAGATION

CONSTANT PROPAGATION

CONSTANT PROPAGATION

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 5 IS:  
 C(142:154,\*,156) C(154:135,\*,157) C(135:134,\*,147) C(142:136,\*,142)  
 C(138:135,\*,138) C(155:136,\*,136) A(2,136,136) A(1,138,138) A(7,142,142)  
 A(6,147,147) A(3,\*,157) A(9,142,142) A(8,142,142) A(5,133,133)

CURRENT POOL FOR BLOCK# 4 IS:  
 A(9,142,142) A(8,142,142) A(7,142,142) A(5,133,133) A(3,\*,151)  
 A(1,138,138) A(2,\*,152)

AFTER PERFORMING THE MEET OPERATION:  
 NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

# FINAL OPTIMIZATION RESULTS

#####





OPTIMIZATION AT 15  
(131,132|15,138) CONSTANT PROPAGATION

OPTIMIZATION AT 22  
(138,131|22,140) CONSTANT PROPAGATION

OPTIMIZATION AT 41  
(131,133|41,142) CONSTANT PROPAGATION

OPTIMIZATION AT 73  
(131,133|41,142) COMM SUBEXP ELIM & CONS PROP

OPTIMIZATION AT 81  
(131,133|41,142) COMM SUBEXP ELIM & CONS PROP

OPTIMIZATION AT 104  
(134,135|104,147) CONSTANT PROPAGATION

OPTIMIZATION AT 111  
(136,142|111,142) CONSTANT PROPAGATION

OPTIMIZATION AT 118  
(135,138|118,138) CONSTANT PROPAGATION

OPTIMIZATION AT 125  
(136,155|125,136) SIMPLIFICATION

```

**** FORMATTED INTERMEDIATE CODE DUMP ****
LOC OFF  OP CODE  CNS ADR  RAW CODE  OPTIMIZATION
  SET                               /ETC      TYPE
*****
0      0      ENT      131      00030083
1      0      TOGGLE   257      00080101
2      0      LOC      2       00020002
3      0      INT      1       00050001
4      0      STO      0       00280000
5      0      DEL      0       002A0000
6      0      LOC      3       00020003
7      0      INT      2       00050002
8      0      STO      0       00280000
9      0      DEL      0       002A0000
10     0      LOC      1       00020001
11     0      LOC      2       00020002
12     0      LOD      0       00270000
13     0      LOC      3       00020003
14     0      LOD      0       00270000
15     0      MUL      C 1024   00148400      CONSTANT PROPAGATION
16     0      STO      0       00280000
17     0      DEL      0       002A0000
18     0      LOC      1       00020001
19     0      LOD      0       00270000
20     0      LOC      2       00020002
21     0      LOD      0       00270000
22     0      GTR      C 1020   002183FC      CONSTANT PROPAGATION
23     0      XIT      46      0004002E
24     0      XIT      26      0004001A
25     0      BSC      0       002F0000
26     0      ENT      135      00030087
27     0      LOC      3       00020003
28     0      INT      3       00050003
29     0      STO      0       00280000
30     0      DEL      0       002A0000
31     0      LOC      5       00020005
32     0      LOC      3       00020003
33     0      LOD      0       00270000
34     0      STO      0       00280000
35     0      DEL      0       002A0000
36     0      LOC      7       00020007
37     0      LOC      2       00020002
38     0      LOD      0       00270000
39     0      LOC      5       00020005
40     0      LOD      0       00270000
41     0      MUL      C 1012   001483F4      CONSTANT PROPAGATION
42     0      STO      0       00280000
43     0      DEL      0       002A0000
44     0      XIT      67      00040043
45     0      BRS      0       002E0000
46     0      ENT      1       00030001
47     0      LOC      2       00020002
48     0      LOC      2       00020002
49     0      LOD      0       00270000
50     0      INT      4       00050004
51     0      ADD      0       00100000
52     0      STO      0       00280000
53     0      DEL      0       002A0000
54     0      LOC      4       00020004
55     0      LOC      2       00020002

```



56	0	LOD	0	00270000	
57	0	STO	0	00280000	
58	0	DEL	0	002A0000	
59	0	LOC	6	00020006	
60	0	LOC	4	00020004	
61	0	LOD	0	00270000	
62	0	LOC	3	00020003	
63	0	LOD	0	00270000	
64	0	MUL	0	00140000	
65	0	STO	0	00280000	
66	0	DEL	0	002A0000	
67	0	ENT	140	0003008C	
68	0	LOC	8	00020008	
69	0	LOC	3	00020003	
70	0	LOD	0	00270000	
71	0	LOC	2	00020002	
72	0	LOD	0	00270000	
73	0	MUL	C 1002	001483EA	COMM SUBEXP ELIM & CONS PROP
74	0	STO	0	00280000	
75	0	DEL	0	002A0000	
76	0	LOC	9	00020009	
77	0	LOC	2	00020002	
78	0	LOD	0	00270000	
79	0	LOC	3	00020003	
80	0	LOD	0	00270000	
81	0	MUL	C 998	001483E6	COMM SUBEXP ELIM & CONS PROP
82	0	STO	0	00280000	
83	0	DEL	0	002A0000	
84	0	ENT	145	00030091	
85	0	LOC	8	00020008	
86	0	LOD	0	00270000	
87	0	LOC	3	00020003	
88	0	LOD	0	00270000	
89	0	GTR	988	002103DC	
90	0	XIT	130	00040082	
91	0	XIT	93	0004005D	
92	0	BSC	0	002F0000	
93	0	ENT	150	00030096	
94	0	LOC	3	00020003	
95	0	LOC	3	00020003	
96	0	LOD	0	00270000	
97	0	INT	5	00050005	
98	0	ADD	976	001003D0	
99	0	STO	0	00280000	
100	0	DEL	0	002A0000	
101	0	LOC	6	00020006	
102	0	INT	5	00050005	
103	0	INT	4	00050004	
104	0	ADD	C 972	001083CC	CONSTANT PROPAGATION
105	0	STO	0	00280000	
106	0	DEL	0	002A0000	
107	0	LOC	7	00020007	
108	0	LOC	7	00020007	
109	0	LOD	0	00270000	
110	0	INT	6	00050006	
111	0	ADD	C 968	001083C8	CONSTANT PROPAGATION
112	0	STO	0	00280000	
113	0	DEL	0	002A0000	
114	0	LOC	1	00020001	
115	0	LOC	1	00020001	
116	0	LOD	0	00270000	
117	0	INT	5	00050005	
118	0	MUL	C 964	001483C4	CONSTANT PROPAGATION
119	0	STO	0	00280000	
120	0	DEL	0	002A0000	
121	0	LOC	2	00020002	
122	0	LOC	2	00020002	
123	0	LOD	0	00270000	
124	0	INT	6	00050006	
125	0	MUL	C 960	001483C0	SIMPLIFICATION
126	0	STO	0	00280000	
127	0	DEL	0	002A0000	
128	0	XIT	84	00040054	
129	0	BRS	0	002E0000	
130	0	ENT	155	00030098	

\*\*\*\*\*

FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

FINAL POOL FOR BLOCK# 1 IS:  
(NULL)



FINAL POCL FOR BLOCK# 2 IS:  
 C(131:132,\*,138) C(138:131,\*,140) A(1,138,138) A(3,132,132) A(2,131,131)

FINAL POCL FOR BLOCK# 3 IS:  
 C(138:131,\*,140) C(131:132,\*,138) C(131:133,\*,142) A(7,142,142)  
 A(5,133,133) A(3,133,133) A(1,138,138) A(2,131,131)

FINAL POCL FOR BLOCK# 4 IS:  
 A(9,142,142) A(8,142,142) A(7,142,142) A(5,133,133) A(3,\*,160)  
 A(1,138,138) A(2,\*,161)

FINAL POCL FOR BLOCK# 5 IS:  
 A(9,142,142) A(8,142,142) A(7,142,142) A(5,133,133) A(3,\*,154)  
 A(1,138,138) A(2,\*,155) C(142:154,\*,156)

FINAL POCL FOR BLOCK# 6 IS:  
 C(142:151,\*,153) A(9,142,142) A(8,142,142) A(7,142,142) A(5,133,133)  
 A(3,\*,151) A(1,138,138) A(2,\*,152)

BASIC BLOCK #1  
 BEGINNING AT 0, ENDING AT 25  
 BLOCK TRAVERSED 1 TIMES  
 0 REFERENCES:  
 BASIC BLOCK #2  
 BEGINNING AT 26, ENDING AT 45  
 BLOCK TRAVERSED 1 TIMES  
 1 REFERENCES:  
 1  
 BASIC BLOCK #3  
 BEGINNING AT 67, ENDING AT 84  
 BLOCK TRAVERSED 1 TIMES  
 1 REFERENCES:  
 2  
 BASIC BLOCK #4  
 BEGINNING AT 84, ENDING AT 92  
 BLOCK TRAVERSED 2 TIMES  
 1 REFERENCES:  
 3  
 BASIC BLOCK #5  
 BEGINNING AT 93, ENDING AT 129  
 BLOCK TRAVERSED 2 TIMES  
 1 REFERENCES:  
 4  
 BASIC BLOCK #6  
 BEGINNING AT 130, ENDING AT 130  
 BLOCK TRAVERSED 1 TIMES  
 1 REFERENCES:  
 4

# BLOCK SUMMARY DATA:

BLOCK NUMBER	NUMBER OF PASSES
1	1
2	1
3	1
4	2
5	2
6	1

TOTAL NUMBER OF BLOCKS PROCESSED WAS 8  
 USING THE FIRST-IN-FIRST-OUT BLOCK SELECTION ALGORITHM.  
 TIME FOR THE OPTIMIZATION WAS 0.270 SECONDS.

\*\*\*CODE SYNTHESIS FILTER IS COMPLETE\*\*\*



CARD	BL	SYL	COMMENT
1	0	0	TEST PROGRAM # 5 ;
3	0	1	\$EXECUTE BEGIN LOCAL A, B, C, D, E, F, G, H, I;
4	1	1	B := 5;
5	1	5	C := 10;
6	1	9	A := B * C;
7	1	17	IF A GTR B THEN
8	1	22	BEGIN
9	1	26	E := C;
10	2	31	G := B * E;
11	2	39	END
12	2	39	ELSE
13	1	39	BEGIN
14	1	42	D := B;
15	2	47	F := D * C;
16	2	55	END;
17	1	56	H := C * B;
18	1	64	IF A GTR B THEN
19	1	69	BEGIN
20	1	73	C := 30;
21	2	77	E := C;
22	2	82	G := B * C;
23	2	90	END
24	2	90	ELSE
25	1	90	BEGIN
26	1	93	B := 20;
27	2	97	D := B;
28	2	102	F := D * C;
29	2	110	END;
30	1	111	H := C * B;
31	1	119	I := B * C;
32	1	127	END
33	1	127	EOF EOF

CODE FILE COPIED (127 WORDS)  
 CONSTANT TABLE COPIED (8 WORDS)  
 2 RECORDS WRITTEN INTO FILE 1  
 END OF CCMPILATION FEBRUARY 12, 1975. CLOCK TIME = 20:31:47.40.

33 CARDS WERE READ.  
 NO ERRORS WERE DETECTED.





\*\*\*CODE SYNTHESIS FILTER\*\*\*

BLOCK SELECTION METHOD WILL BE LAST-IN-FIRST-OUT

\*\*\*C O M P L E T E T A B L E D U M P\*\*\*  
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 0

CONSTANTS	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
CONSYM	5	10	30	20										
CCNTYPE	INT	INT	INT	INT										
CONVAL	128	129	130	131										
CONINT	5	10	30	20										

ADDR TABL	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
(EMPTY)														

VALUE STACK (TOP AT 127)

CONT STK	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
BLOCK#	0													
ENTRY	0													

EXEC STAC	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
(EMPTY)														

\*\*\*\* FORMATTED INTERMEDIATE CODE DUMP \*\*\*\*

LOC	OFF	OP CODE	CNS	ADR	RAW CODE	OPTIMIZATION
SET			/ETC			TYPE
0	0	ENT	C	128	00038080	
1	0	TOGGLE		257	000B0101	
2	0	LOC		2	00020002	
3	0	INT		1	00050001	
4	0	STO		0	00280000	
5	0	DEL		0	002A0000	
6	0	LOC		3	00020003	
7	0	INT		2	00050002	
8	0	STO		0	00280000	
9	0	DEL		0	002A0000	
10	0	LOC		1	00020001	
11	0	LOC		2	00020002	
12	0	LOD		0	00270000	
13	0	LOC		3	00020003	
14	0	LOD		0	00270000	
15	0	MUL		0	00140000	
16	0	STO		0	00280000	
17	0	DEL		0	002A0000	
18	0	LOC		1	00020001	
19	0	LOD		0	00270000	
20	0	LOC		2	00020002	
21	0	LOD		0	00270000	
22	0	GTR		0	00210000	
23	0	XIT		42	0004002A	
24	0	XIT		26	0004001A	
25	0	BSC		0	002F0000	
26	0	ENT		1	00030001	
27	0	LOC		5	00020005	
28	0	LOC		3	00020003	
29	0	LOD		0	00270000	
30	0	STO		0	00280000	
31	0	DEL		0	002A0000	
32	0	LOC		7	00020007	
33	0	LOC		2	00020002	
34	0	LOD		0	00270000	
35	0	LOC		5	00020005	
36	0	LOD		0	00270000	
37	0	MUL		0	00140000	
38	0	STO		0	00280000	
39	0	DEL		0	002A0000	
40	0	XIT		56	00040038	
41	0	BRS		0	002E0000	



42	0	END	1	00030001
43	0	LOC	4	00020004
44	0	LOC	2	00020002
45	0	LOC	0	00270000
46	0	STO	0	00280000
47	0	DEL	0	002A0000
48	0	LOC	6	00020006
49	0	LOC	4	00020004
50	0	LOC	0	00270000
51	0	LOC	3	00020003
52	0	LOC	0	00270000
53	0	MUL	0	00140000
54	0	STO	0	00280000
55	0	DEL	0	002A0000
56	0	ENT	1	00030001
57	0	LOC	8	00020008
58	0	LOC	3	00020003
59	0	LOC	0	00270000
60	0	LOC	2	00020002
61	0	LOC	0	00270000
62	0	MUL	0	00140000
63	0	STO	0	00280000
64	0	DEL	0	002A0000
65	0	LOC	1	00020001
66	0	LOC	0	00270000
67	0	LOC	2	00020002
68	0	LOC	0	00270000
69	0	GTR	0	00210000
70	0	XIT	93	0004005D
71	0	XIT	73	00040049
72	0	BSC	0	002F0000
73	0	ENT	1	00030001
74	0	LOC	3	00020003
75	0	INT	3	00050003
76	0	STO	0	00280000
77	0	DEL	0	002A0000
78	0	LOC	5	00020005
79	0	LOC	3	00020003
80	0	LOC	0	00270000
81	0	STO	0	00280000
82	0	DEL	0	002A0000
83	0	LOC	7	00020007
84	0	LOC	2	00020002
85	0	LOC	0	00270000
86	0	LOC	3	00020003
87	0	LOC	0	00270000
88	0	MUL	0	00140000
89	0	STO	0	00280000
90	0	DEL	0	002A0000
91	0	XIT	111	0004006F
92	0	BRS	0	002E0000
93	0	ENT	1	00030001
94	0	LOC	2	00020002
95	0	INT	4	00050004
96	0	STO	0	00280000
97	0	DEL	0	002A0000
98	0	LOC	4	00020004
99	0	LOC	2	00020002
100	0	LOC	0	00270000
101	0	STO	0	00280000
102	0	DEL	0	002A0000
103	0	LOC	6	00020006
104	0	LOC	4	00020004
105	0	LOC	0	00270000
106	0	LOC	3	00020003
107	0	LOC	0	00270000
108	0	MUL	0	00140000
109	0	STO	0	00280000
110	0	DEL	0	002A0000
111	0	ENT	1	00030001
112	0	LOC	8	00020008
113	0	LOC	3	00020003
114	0	LOC	0	00270000
115	0	LOC	2	00020002
116	0	LOC	0	00270000
117	0	MUL	0	00140000
118	0	STO	0	00280000
119	0	DEL	0	002A0000
120	0	LOC	9	00020009
121	0	LOC	2	00020002
122	0	LOC	0	00270000
123	0	LOC	3	00020003
124	0	LOC	0	00270000
125	0	MUL	0	00140000
126	0	STO	0	00280000
127	0	DEL	0	002A0000

\*\*\*\*\*



BASIC BLOCK #1  
 BEGINNING AT 0, ENDING AT 0  
 BLOCK TRAVERSED 0 TIMES  
 0 REFERENCES:

\*\*\*END OF COMPLETE TABLE DUMP\*\*\*

B+2	0	LOC	2	00020002
B+3	0	INT	1	00050001
B+4	0	STO	0	00280000
B+5	0	DEL	0	002A0000
B+6	0	LOC	3	00020003
B+7	0	INT	2	00050002
B+8	0	STO	0	00280000
B+9	0	DEL	0	002A0000
B+10	0	LOC	1	00020001
B+11	0	LOC	2	00020002
B+12	0	LOD	0	00270000
B+13	0	LOC	3	00020003
B+14	0	LOD	0	00270000
B+15	0	MUL	0	00140000
B+16	0	STO	0	00280000
B+17	0	DEL	0	002A0000
B+18	0	LOC	1	00020001
B+19	0	LOD	0	00270000
B+20	0	LOC	2	00020002
B+21	0	LOD	0	00270000
B+22	0	GTR	0	00210000
B+23	0	XIT	42	0004002A
B+24	0	XIT	26	0004001A
B+25	0	BSC	0	002F0000

INITIAL CURRENT OPTIMIZED PCOL TO BE USED FOR BLOCK# 2 IS:  
 C(128:129,\*,133) C(133:128,\*,135) A(1,133,133) A(3,129,129) A(2,128,128)

CONTROL AT 26, FROM BLK 1 TO BLK 2 (PASS 1)

C+26	0	ENT	C 132	00038084
B+27	0	LOC	5	00020005
B+28	0	LOC	3	00020003
B+29	0	LOD	0	00270000
B+30	0	STO	0	00280000
B+31	0	DEL	0	002A0000
B+32	0	LOC	7	00020007
B+33	0	LOC	2	00020002
B+34	0	LOD	0	00270000
B+35	0	LOC	5	00020005
B+36	0	LOD	0	00270000
B+37	0	MUL	0	00140000
B+38	0	STO	0	00280000
B+39	0	DEL	0	002A0000
B+40	0	XIT	56	00040038
B+41	0	BRS	0	002E0000

INITIAL CURRENT OPTIMIZED PCOL TO BE USED FOR BLOCK# 3 IS:  
 C(133:128,\*,135) C(128:129,\*,133) A(7,133,133) A(5,129,129) A(1,133,133)  
 A(3,129,129) A(2,128,128)

CONTROL AT 56, FROM BLK 2 TO BLK 3 (PASS 1)

C+56	0	ENT	C 137	00038089
B+57	0	LOC	8	00020008
B+58	0	LOC	3	00020003
B+59	0	LOD	0	00270000
B+60	0	LOC	2	00020002
B+61	0	LOD	0	00270000
B+62	0	MUL	0	00140000
B+63	0	STO	0	00280000
B+64	0	DEL	0	002A0000
B+65	0	LOC	1	00020001
B+66	0	LOD	0	00270000
B+67	0	LOC	2	00020002
B+68	0	LOD	0	00270000
B+69	0	GTR	0	00210000
B+70	0	XIT	93	00040050
B+71	0	XIT	73	00040049
B+72	0	BSC	0	002F0000

INITIAL CURRENT OPTIMIZED PCOL TO BE USED FOR BLOCK# 4 IS:  
 C(128:129,\*,133) C(133:128,\*,135) A(8,133,133) A(7,133,133) A(5,129,129)  
 A(1,133,133) A(3,129,129) A(2,128,128)

CONTROL AT 73, FROM BLK 3 TO BLK 4 (PASS 1)

C+73	0	ENT	C 142	0003808E
B+74	0	LOC	3	00020003
B+75	0	INT	3	00050003
B+76	0	STO	0	00280000
B+77	0	DEL	0	002A0000
B+78	0	LOC	5	00020005





B+79	0	LUC	3	00020003
B+80	0	LCD	0	00270000
B+81	0	STO	0	00280000
B+82	0	DEL	0	002A0000
B+83	0	LOC	7	00020007
B+84	0	LOC	2	00020002
B+85	0	LOD	0	00270000
B+86	0	LOC	3	00020003
B+87	0	LOD	0	00270000
B+88	0	MUL	0	00140000
B+89	0	STO	0	00280000
B+90	0	DEL	0	002A0000
B+91	0	XIT	111	0004006F
B+92	0	BRS	0	002E0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 5 IS:  
 C(133:128,\*,135) C(128:129,\*,133) C(128:130,\*,137) A(7,137,137)  
 A(5,130,130) A(3,130,130) A(8,133,133) A(1,133,133) A(2,128,128)

CONTROL AT 111, FROM BLK 4 TO BLK 5 (PASS 1)				
C+111	0	ENT	C 147	00038093
B+112	0	LOC	8	00020008
B+113	0	LOC	3	00020003
B+114	0	LOD	0	00270000
B+115	0	LOC	2	00020002
B+116	0	LOD	0	00270000
B+117	0	MUL	0	00140000
B+118	0	STO	0	00280000
B+119	0	DEL	0	002A0000
B+120	0	LOC	9	00020009
B+121	0	LOC	2	00020002
B+122	0	LOD	0	00270000
B+123	0	LOC	3	00020003
B+124	0	LOD	0	00270000
B+125	0	MUL	0	00140000
B+126	0	STO	0	00280000
B+127	0	DEL	0	002A0000

#### FINAL OPTIMIZATION RESULTS

#####

OPTIMIZATION AT 15	(128,129 15,133)	CONSTANT PROPAGATION
OPTIMIZATION AT 22	(133,128 22,135)	CONSTANT PROPAGATION
OPTIMIZATION AT 37	(128,129 15,133)	COMM SUBEXP ELIM & CONS PROP
OPTIMIZATION AT 62	(128,129 15,133)	COMM SUBEXP ELIM & CONS PROP
OPTIMIZATION AT 69	(133,128 22,135)	COMM SUBEXP ELIM & CONS PROP
OPTIMIZATION AT 88	(130,128 88,137)	CONSTANT PROPAGATION
OPTIMIZATION AT 117	(130,128 88,137)	COMM SUBEXP ELIM & CONS PROP
OPTIMIZATION AT 125	(130,128 88,137)	COMM SUBEXP ELIM & CONS PROP

**** FORMATTED INTERMEDIATE CODE DUMP ****					
LOC	OFF	OP CODE	CNS ADR	RAW CODE	OPTIMIZATION TYPE
			SET	/ETC	
0	0	ENT	128	00030080	
1	0	TOGGLE	257	00030101	
2	0	LOC	2	00020002	
3	0	INT	1	00050001	
4	0	STO	0	00280000	
5	0	DEL	0	002A0000	
6	0	LOC	3	00020003	
7	0	INT	2	00050002	
8	0	STO	0	00280000	
9	0	DEL	0	002A0000	
10	0	LOC	1	00020001	
11	0	LOC	2	00020002	
12	0	LOD	0	00270000	
13	0	LOC	3	00020003	
14	0	LOD	0	00270000	





15	0	MUL	C	1024	00148400	CONSTANT PROPAGATION
16	0	STO		0	00280000	
17	0	DEL		0	002A0000	
18	0	LOC		1	00020001	
19	0	LOD		0	00270000	
20	0	LOC		2	00020002	
21	0	LOD		0	00270000	
22	0	GTR	C	1020	002183FC	CONSTANT PROPAGATION
23	0	XIT		42	0004002A	
24	0	XIT		26	0004001A	
25	0	BSC		0	002F0000	
26	0	ENT		132	00030084	
27	0	LOC		5	00020005	
28	0	LOC		3	00020003	
29	0	LOD		0	00270000	
30	0	STO		0	00280000	
31	0	DEL		0	002A0000	
32	0	LOC		7	00020007	
33	0	LOC		2	00020002	
34	0	LOD		0	00270000	
35	0	LOC		5	00020005	
36	0	LOD		0	00270000	
37	0	MUL	C	1012	001483F4	COMM SUBEXP ELIM & CONS PROP
38	0	STO		0	00280000	
39	0	DEL		0	002A0000	
40	0	XIT		56	00040038	
41	0	BRS		0	002E0000	
42	0	ENT		1	00030001	
43	0	LOC		4	00020004	
44	0	LOC		2	00020002	
45	0	LOD		0	00270000	
46	0	STO		0	00280000	
47	0	DEL		0	002A0000	
48	0	LOC		6	00020006	
49	0	LOC		4	00020004	
50	0	LOD		0	00270000	
51	0	LOC		3	00020003	
52	0	LOD		0	00270000	
53	0	MUL		0	00140030	
54	0	STO		0	00280000	
55	0	DEL		0	002A0000	
56	0	ENT		137	00030069	
57	0	LOC		8	00020008	
58	0	LOC		3	00020003	
59	0	LOD		0	00270000	
60	0	LOC		2	00020002	
61	0	LOD		0	00270000	
62	0	MUL	C	1004	001483EC	COMM SUBEXP ELIM & CONS PROP
63	0	STO		0	00280000	
64	0	DEL		0	002A0000	
65	0	LOC		1	00020001	
66	0	LOD		0	00270000	
67	0	LOC		2	00020002	
68	0	LOD		0	00270000	
69	0	GTR	C	1000	002183E8	COMM SUBEXP ELIM & CONS PROP
70	0	XIT		93	0004005D	
71	0	XIT		73	00040049	
72	0	BSC		0	002F0000	
73	0	ENT		142	0003008E	
74	0	LOC		3	00020003	
75	0	INT		3	00050003	
76	0	STO		0	00280000	
77	0	DEL		0	002A0000	
78	0	LOC		5	00020005	
79	0	LOC		3	00020003	
80	0	LOD		0	00270000	
81	0	STO		0	00280000	
82	0	DEL		0	002A0000	
83	0	LOC		7	00020007	
84	0	LOC		2	00020002	
85	0	LOD		0	00270000	
86	0	LOC		3	00020003	
87	0	LOD		0	00270000	
88	0	MUL	C	992	001483E0	CONSTANT PROPAGATION
89	0	STO		0	00280000	
90	0	DEL		0	002A0000	
91	0	XIT		111	0004006F	
92	0	BRS		0	002E0000	
93	0	ENT		1	00030001	
94	0	LOC		2	00020002	
95	0	INT		4	00050004	
96	0	STO		0	00280000	
97	0	DEL		0	002A0000	
98	0	LOC		4	00020004	
99	0	LOC		2	00020002	
100	0	LOD		0	00270000	
101	0	STO		0	00280000	
102	0	DEL		0	002A0000	



103	0	LDC	6	00020006	
104	0	LOC	4	00020004	
105	0	LOD	0	00270000	
106	0	LOC	3	00020003	
107	0	LOD	0	00270000	
108	0	MUL	0	00140000	
109	0	STO	0	00280000	
110	0	DEL	0	002A0000	
111	0	ENT	147	00030093	
112	0	LOC	8	00020008	
113	0	LOC	3	00020003	
114	0	LOD	0	00270000	
115	0	LOC	2	00020002	
116	0	LOD	0	00270000	
117	0	MUL	C 982	001483D6	COMM SUBEXP ELIM & CONS PROP
118	0	STO	0	00280000	
119	0	DEL	0	002A0000	
120	0	LOC	9	00020009	
121	0	LOC	2	00020002	
122	0	LOD	0	00270000	
123	0	LOC	3	00020003	
124	0	LCD	0	00270000	
125	0	MUL	C 978	001483D2	COMM SUBEXP ELIM & CONS PROP
126	0	STO	0	00280000	
127	0	DEL	0	002A0000	

\*\*\*\*\*

FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

FINAL POCL FOR BLOCK# 1 IS:  
(NULL)

FINAL POCL FOR BLOCK# 2 IS:  
C(128:129,\*,133) C(133:128,\*,135) A(1,133,133) A(3,129,129) A(2,128,128)

FINAL POCL FOR BLOCK# 3 IS:  
C(133:128,\*,135) C(128:129,\*,133) A(7,133,133) A(5,129,129) A(1,133,133)  
A(3,129,129) A(2,128,128)

FINAL POCL FOR BLOCK# 4 IS:  
C(128:129,\*,133) C(133:128,\*,135) A(8,133,133) A(7,133,133) A(5,129,129)  
A(1,133,133) A(3,129,129) A(2,128,128)

FINAL POCL FOR BLOCK# 5 IS:  
C(133:128,\*,135) C(128:129,\*,133) C(128:130,\*,137) A(7,137,137)  
A(5,130,130) A(3,130,130) A(8,133,133) A(1,133,133) A(2,128,128)

BASIC BLOCK #1  
BEGINNING AT 0, ENDING AT 25  
BLOCK TRAVERSED 1 TIMES  
0 REFERENCES:

BASIC BLOCK #2  
BEGINNING AT 26, ENDING AT 41  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:

1  
BASIC BLOCK #3  
BEGINNING AT 56, ENDING AT 72  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:

2  
BASIC BLOCK #4  
BEGINNING AT 73, ENDING AT 92  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:

3  
BASIC BLOCK #5  
BEGINNING AT 111, ENDING AT 127  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:

4

BLOCK SUMMARY DATA:



BLOCK NUMBER

NUMBER OF PASSES

1  
2  
3  
4  
5

1  
1  
1  
1  
1

TOTAL NUMBER OF BLOCKS PROCESSED WAS 5  
USING THE LAST-IN-FIRST-OUT BLOCK SELECTION ALGORITHM.  
TIME FOR THE OPTIMIZATION WAS 0.170 SECONDS.

\*\*\*CODE SYNTHESIS FILTER IS COMPLETE\*\*\*



CARD	BL	SYL	COMMENT
1	0	0	TEST PROGRAM # 6 ;
3	0	1	\$EXECUTE BEGIN LOCAL A,B,C,D,E,F,G,H,I,J;
4	1	1	READ(A, B);
5	1	7	WHILE A GTR B DO
6	1	17	BEGIN
7	1	17	IF A GTR B THEN
8	2	22	BEGIN
9	2	26	IF A GTR B THEN
10	3	31	BEGIN
11	3	35	B := 10;
12	4	39	A := B * C;
13	4	47	END
14	4	47	ELSE
15	3	47	BEGIN
16	3	50	B := 20;
17	4	54	D := B;
18	4	59	F := D * C;
19	4	67	END;
20	3	68	E := B;
21	3	73	H := E * C;
22	3	81	END
23	3	81	ELSE
24	2	81	BEGIN
25	2	84	B := 30;
26	3	88	J := C * B;
27	3	96	END;
28	2	97	F := C;
29	2	102	I := B * C;
30	2	110	END;
31	1	113	END
32	1	113	EOF

CODE FILE COPIED (113 WORDS)  
 CONSTANT TABLE COPIED (6 WORDS)  
 2 RECORDS WRITTEN INTO FILE 1  
 END OF COMPILATION FEBRUARY 12, 1975. CLOCK TIME = 20:42:29.46.

32 CARDS WERE READ.  
 NO ERRORS WERE DETECTED.





\*\*\*CODE SYNTHESIS FILTER\*\*\*

BLOCK SELECTION METHOD WILL BE LAST-IN-FIRST-OUT

\*\*\*C O M P L E T E T A B L E D U M P\*\*\*  
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 0

CONSTANTS	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
CONSYM	10	20	30											
CCNTYPE	INT	INT	INT											
CONVAL	114	115	116											
CONINT	10	20	30											

ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

VALUE STACK (TOP AT 127)

CONT STK	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
BLOCK#	0													
ENTRY	0													

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

\*\*\*\* FORMATTED INTERMEDIATE CODE DUMP \*\*\*\*

LOC	OFF	OP CODE	CNS	ADR	RAW CODE	OPTIMIZATION
SET			/ETC			TYPE
0	0	ENT	C	114	00038072	
1	0	TOGGLE		257	00080101	
2	0	LOC		1	00020001	
3	0	INT		0	00050000	
4	0	STD		0	00290000	
5	0	LOC		2	00020002	
6	0	INT		0	00050000	
7	0	STD		0	00290000	
8	0	ENT		1	00030001	
9	0	LOC		1	00020001	
10	0	LOD		0	00270000	
11	0	LOC		2	00020002	
12	0	LOD		0	00270000	
13	0	GTR		0	00210000	
14	0	XIT		113	00040071	
15	0	XIT		17	00040011	
16	0	BSC		0	002F0000	
17	0	ENT		1	00030001	
18	0	LOC		1	00020001	
19	0	LOD		0	00270000	
20	0	LOC		2	00020002	
21	0	LOD		0	00270000	
22	0	GTR		0	00210000	
23	0	XIT		84	00040054	
24	0	XIT		26	0004001A	
25	0	BSC		0	002F0000	
26	0	ENT		1	00030001	
27	0	LOC		1	00020001	
28	0	LOD		0	00270000	
29	0	LOC		2	00020002	
30	0	LOD		0	00270000	
31	0	GTR		0	00210000	
32	0	XIT		50	00040032	
33	0	XIT		35	00040023	
34	0	BSC		0	002F0000	
35	0	ENT		1	00030001	
36	0	LOC		2	00020002	
37	0	INT		1	00050001	
38	0	STO		0	00280000	
39	0	DEL		0	002A0000	
40	0	LOC		1	00020001	
41	0	LOC		2	00020002	



42	0	LOD	0	00270000
43	0	LOC	3	00020003
44	0	LOD	0	00270000
45	0	MUL	0	00140000
46	0	STO	0	00280000
47	0	DEL	0	002A0000
48	0	XIT	68	00040044
49	0	BRS	0	002E0000
50	0	ENT	1	00030001
51	0	LOC	2	00020002
52	0	INT	2	00050002
53	0	STO	0	00280000
54	0	DEL	0	002A0000
55	0	LOC	4	00020004
56	0	LOC	2	00020002
57	0	LOD	0	00270000
58	0	STO	0	00280000
59	0	DEL	0	002A0000
60	0	LOC	6	00020006
61	0	LOC	4	00020004
62	0	LOD	0	00270000
63	0	LOC	3	00020003
64	0	LOD	0	00270000
65	0	MUL	0	00140000
66	0	STO	0	00280000
67	0	DEL	0	002A0000
68	0	ENT	1	00030001
69	0	LOC	5	00020005
70	0	LOC	2	00020002
71	0	LOD	0	00270000
72	0	STO	0	00280000
73	0	DEL	0	002A0000
74	0	LOC	8	00020008
75	0	LOC	5	00020005
76	0	LOD	0	00270000
77	0	LOC	3	00020003
78	0	LOD	0	00270000
79	0	MUL	0	00140000
80	0	STO	0	00280000
81	0	DEL	0	002A0000
82	0	XIT	97	00040061
83	0	BRS	0	002E0000
84	0	ENT	1	00030001
85	0	LOC	2	00020002
86	0	INT	3	00050003
87	0	STO	0	00280000
88	0	DEL	0	002A0000
89	0	LOC	10	0002000A
90	0	LOC	3	00020003
91	0	LOD	0	00270000
92	0	LOC	2	00020002
93	0	LOD	0	00270000
94	0	MUL	0	00140000
95	0	STO	0	00280000
96	0	DEL	0	002A0000
97	0	ENT	1	00030001
98	0	LOC	5	00020005
99	0	LOC	3	00020003
100	0	LOD	0	00270000
101	0	STO	0	00280000
102	0	DEL	0	002A0000
103	0	LOC	9	00020009
104	0	LOC	2	00020002
105	0	LOD	0	00270000
106	0	LOC	3	00020003
107	0	LOD	0	00270000
108	0	MUL	0	00140000
109	0	STO	0	00280000
110	0	DEL	0	002A0000
111	0	XIT	8	00040008
112	0	BRS	0	002E0000
113	0	ENT	1	00030001

\*\*\*\*\*

BASIC BLOCK #1  
 BEGINNING AT 0, ENDING AT 0  
 BLOCK TRAVERSED 0 TIMES  
 0 REFERENCES:

\*\*\*END OF COMPLETE TABLE DUMP\*\*\*

B+2	0	LOC	1	00020001
B+3	0	INT	0	00050000
B+4	0	STO	0	00290000
B+5	0	LOC	2	00020002
B+6	0	INT	0	00050000
B+7	0	STO	0	00290000



B+B 0 IENI 1 11 1 00030001 1

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 2 IS:  
A(2,\*,118) A(1,\*,117)

CONTROL AT 8, FROM BLK 1 TO BLK 2 (PASS 1)			
C+8	0	ENT	C 118 00038076
B+9	0	LOC	1 00020001
B+10	0	LOD	0 00270000
B+11	0	LOC	2 00020002
B+12	0	LOD	0 00270000
B+13	0	GTR	0 00210000
B+14	0	XIT	113 00040071
B+15	0	XIT	17 00040011
B+16	0	BSC	0 002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 3 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 4 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

CONTROL AT 17, FROM BLK 2 TO BLK 4 (PASS 1)			
C+17	0	ENT	C 128 00038080
B+18	0	LOC	1 00020001
B+19	0	LOD	0 00270000
B+20	0	LOC	2 00020002
B+21	0	LOD	0 00270000
B+22	0	GTR	0 00210000
B+23	0	XIT	84 00040054
B+24	0	XIT	26 0004001A
B+25	0	BSC	0 002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 5 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 6 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

CONTROL AT 26, FROM BLK 4 TO BLK 6 (PASS 1)			
C+26	0	ENT	C 138 0003808A
B+27	0	LOC	1 00020001
B+28	0	LOD	0 00270000
B+29	0	LOC	2 00020002
B+30	0	LOD	0 00270000
B+31	0	GTR	0 00210000
B+32	0	XIT	50 00040032
B+33	0	XIT	35 00040023
B+34	0	BSC	0 002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 7 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 8 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

CONTROL AT 35, FROM BLK 6 TO BLK 8 (PASS 1)			
C+35	0	ENT	C 148 00038094
B+36	0	LOC	2 00020002
B+37	0	INT	1 00050001
B+38	0	STO	0 00280000
B+39	0	DEL	0 002A0000
B+40	0	LOC	1 00020001
B+41	0	LOC	2 00020002
B+42	0	LOD	0 00270000
B+43	0	LOC	3 00020003
B+44	0	LOD	0 00270000
B+45	0	MUL	0 00140000
B+46	0	STO	0 00280000
B+47	0	DEL	0 002A0000
B+48	0	XIT	68 00040044
B+49	0	BRS	0 002E0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 9 IS:  
C(117:118,\*,119) C(114:120,\*,121) A(1,\*,121) A(3,\*,120) A(2,114,114)

CONTROL AT 68, FROM BLK 8 TO BLK 9 (PASS 1)			
C+68	0	ENT	C 153 00038099
B+69	0	LOC	5 00020005
B+70	0	LOC	2 00020002
B+71	0	LOD	0 00270000
B+72	0	STO	0 00280000
B+73	0	DEL	0 002A0000
B+74	0	LOC	8 00020008
B+75	0	LOC	5 00020005





B+76	0	LOD	0	00270000
B+77	0	LOC	3	00020003
B+78	0	LOD	0	00270000
B+79	0	MUL	0	00140000
B+80	0	STO	0	00280000
B+81	0	DEL	0	002A0000
B+82	0	XIT	97	00040061
B+83	0	BRS	0	002E0000

INITIAL CURRENT OPTIMIZED PCOL TO BE USED FOR BLOCK# 10 IS:  
 C(114:120,\*,121) C(117:118,\*,119) A(8,\*,121) A(5,114,114) A(1,\*,121)  
 A(3,\*,120) A(2,114,114)

CONTROL AT 97, FROM BLK 9 TO BLK 10 (PASS 1)				
C+97	0	ENT	C 158	0003809E
B+98	0	LOC	5	00020005
B+99	0	LOC	3	00020003
B+100	0	LOD	0	00270000
B+101	0	STO	0	00280000
B+102	0	DEL	0	002A0000
B+103	0	LOC	9	00020009
B+104	0	LOC	2	00020002
B+105	0	LOD	0	00270000
B+106	0	LOC	3	00020003
B+107	0	LOD	0	00270000
B+108	0	MUL	0	00140000
B+109	0	STO	0	00280000
B+110	0	DEL	0	002A0000
B+111	0	XIT	8	00040008
B+112	0	BRS	0	002E0000

BEFORE THE MEET OPERATION:

OUTPUT PCOL FROM BLOCK# 10 IS:  
 C(117:118,\*,119) C(114:120,\*,121) A(9,\*,121) A(5,\*,120) A(8,\*,121)  
 A(1,\*,121) A(3,\*,120) A(2,114,114)

CURRENT POOL FOR BLOCK# 2 IS:  
 A(2,\*,118) A(1,\*,117)

AFTER PERFORMING THE MEET OPERATION:  
 NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

CONTROL AT 50, FROM BLK 6 TO BLK 7 (PASS 1)				
C+50	0	ENT	C 143	0003808F
B+51	0	LOC	2	00020002
B+52	0	INT	2	00050002
B+53	0	STO	0	00280000
B+54	0	DEL	0	002A0000
B+55	0	LOC	4	00020004
B+56	0	LOC	2	00020002
B+57	0	LOD	0	00270000
B+58	0	STO	0	00280000
B+59	0	DEL	0	002A0000
B+60	0	LOC	6	00020006
B+61	0	LOC	4	00020004
B+62	0	LOD	0	00270000
B+63	0	LOC	3	00020003
B+64	0	LOD	0	00270000
B+65	0	MUL	0	00140000
B+66	0	STO	0	00280000
B+67	0	DEL	0	002A0000
B+68	0	ENT	C 153	00038099

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 7 IS:  
 C(117:118,\*,119) C(115:124,\*,125) A(3,\*,124) A(6,\*,125) A(4,115,115)  
 A(2,115,115) A(1,\*,117)

CURRENT POOL FOR BLOCK# 9 IS:  
 C(117:118,\*,119) C(114:120,\*,121) A(1,\*,121) A(3,\*,120) A(2,114,114)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 9 IS:  
 A(1,\*,126) A(3,\*,127) A(2,\*,128) C(128:127,\*,129)

CONTROL AT 68, FROM BLK 7 TO BLK 9 (PASS 2)				
C+68	0	ENT	C 153	00038099
B+69	0	LOC	5	00020005
B+70	0	LOC	2	00020002
B+71	0	LOD	0	00270000
B+72	0	STO	0	00280000
B+73	0	DEL	0	002A0000





B+74	0	LUC	8	00020008	
B+75	0	LOC	5	00020005	
B+76	0	LOD	0	00270000	
B+77	0	LOC	3	00020003	
B+78	0	LOD	0	00270000	
B+79	0	MUL	C 992	001483E0	COMM SUBEXP ELIM
B+80	0	STO	0	00280000	
B+81	0	DEL	0	002A0000	
B+82	0	XIT	97	00040061	
B+83	0	BRS	0	002E0000	

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 9 IS:

C(128:127,\*,129) A(8,\*,129) A(5,\*,128) A(1,\*,126) A(3,\*,127) A(2,\*,128)

CURRENT POOL FOR BLOCK# 10 IS:

C(114:120,\*,121) C(117:118,\*,119) A(8,\*,121) A(5,114,114) A(1,\*,121)  
A(3,\*,120) A(2,114,114)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 10 IS:

A(8,\*,130) A(5,\*,131) A(1,\*,132) A(3,\*,133) A(2,\*,131)

CONTROL AT 97, FROM BLK 9 TO BLK 10 (PASS 2)

C+97	0	ENT	C 158	0003809E	
B+98	0	LOC	5	00020005	
B+99	0	LOC	3	00020003	
B+100	0	LOD	0	00270000	
B+101	0	STO	0	00280000	
B+102	0	DEL	0	002A0000	
B+103	0	LOC	9	00020009	
B+104	0	LOC	2	00020002	
B+105	0	LOD	0	00270000	
B+106	0	LOC	3	00020003	
B+107	0	LOD	0	00270000	
B+108	0	MUL	C 984	001483D8	COMM SUBEXP ELIM
B+109	0	STO	0	00280000	
B+110	0	DEL	0	002A0000	
B+111	0	XIT	8	00040008	
B+112	0	BRS	0	002E0000	

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 10 IS:

C(131:133,\*,134) A(9,\*,134) A(5,\*,133) A(8,\*,130) A(1,\*,132) A(3,\*,133)  
A(2,\*,131)

CURRENT POOL FOR BLOCK# 2 IS:

A(2,\*,122) A(1,\*,123)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

CONTROL AT 84, FROM BLK 4 TO BLK 5 (PASS 1)

C+84	0	ENT	C 133	00038085	
B+85	0	LOC	2	00020002	
B+86	0	INT	3	00050003	
B+87	0	STO	0	00280000	
B+88	0	DEL	0	002A0000	
B+89	0	LOC	10	0002000A	
B+90	0	LOC	3	00020003	
B+91	0	LOD	0	00270000	
B+92	0	LOC	2	00020002	
B+93	0	LOD	0	00270000	
B+94	0	MUL	0	00140000	
B+95	0	STO	0	00280000	
B+96	0	DEL	0	002A0000	
B+97	0	ENT	C 158	0003809E	

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 5 IS:

C(117:118,\*,119) C(137:116,\*,138) A(3,\*,137) A(10,\*,138) A(2,116,116)  
A(1,\*,117)

CURRENT POOL FOR BLOCK# 10 IS:

A(8,\*,130) A(5,\*,131) A(1,\*,132) A(3,\*,133) A(2,\*,131)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 10 IS:

A(1,\*,139) A(3,\*,140) A(2,\*,141)



CONTROL AT 97, FROM BLK 5 TO BLK 10 (PASS 3)			
C+97	0	ENT	C 158 0003809E
B+98	0	LOC	5 00020005
B+99	0	LOC	3 00020003
B+100	0	LOD	0 00270000
B+101	0	STO	0 00280000
B+102	0	DEL	0 002A0000
B+103	0	LOC	9 00020009
B+104	0	LOC	2 00020002
B+105	0	LOD	0 00270000
B+106	0	LOC	3 00020003
B+107	0	LOD	0 00270000
B+108	0	MUL	C 984 001483D8
B+109	0	STO	0 00280000
B+110	0	DEL	0 002A0000
B+111	0	XIT	8 00040008
B+112	0	BRS	0 002E0000

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 10 IS:  
C(141:140,\*,142) A(9,\*,142) A(5,\*,140) A(1,\*,139) A(3,\*,140) A(2,\*,141)

CURRENT POOL FOR BLOCK# 2 IS:  
A(2,\*,135) A(1,\*,136)

AFTER PERFORMING THE MEET OPERATION:  
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

CONTROL AT 113, FROM BLK 2 TO BLK 3 (PASS 1)			
C+113	0	ENT	C 123 0003807B

# FINAL OPTIMIZATION RESULTS

OPTIMIZATION AT 22  
(117,118|13,119) COMM SUBEXP ELIM

OPTIMIZATION AT 31  
(117,118|13,119) COMM SUBEXP ELIM

OPTIMIZATION AT 79  
(127,128|45,129) COMM SUBEXP ELIM

**** FORMATTED INTERMEDIATE CODE DUMP ****					OPTIMIZATION TYPE
LOC	OFF SET	OP CODE	CNS ADR /ETC	RAW CODE	
0	0	ENT	114	00030072	
1	0	TOGGLE	257	00080101	
2	0	LOC	1	00020001	
3	0	INT	0	00050000	
4	0	STO	0	00290000	
5	0	LOC	2	00020002	
6	0	INT	0	00050000	
7	0	STO	0	00290000	
8	0	ENT	118	00030076	
9	0	LOC	1	00020001	
10	0	LOD	0	00270000	
11	0	LOC	2	00020002	
12	0	LOD	0	00270000	
13	0	GTR	1024	00210400	
14	0	XIT	113	00040071	
15	0	XIT	17	00040011	
16	0	BSC	0	002F0000	
17	0	ENT	128	00030080	
18	0	LOC	1	00020001	
19	0	LOD	0	00270000	
20	0	LOC	2	00020002	
21	0	LOD	0	00270000	
22	0	GTR	C 1016	002183F8	COMM SUBEXP ELIM
23	0	XIT	84	00040054	
24	0	XIT	26	0004001A	
25	0	BSC	0	002F0000	
26	0	ENT	138	0003008A	
27	0	LOC	1	00020001	
28	0	LOD	0	00270000	
29	0	LOC	2	00020002	
30	0	LOD	0	00270000	
31	0	GTR	C 1008	002183F0	COMM SUBEXP ELIM
32	0	XIT	50	00040032	



33	U	XIT	35	00040023
34	0	BSC	0	002F0000
35	0	ENT	148	00030094
36	0	LOC	2	00020002
37	0	INT	1	00050001
38	0	STO	0	00280000
39	0	DEL	0	002A0000
40	0	LOC	1	00020001
41	0	LOC	2	00020002
42	0	LOC	0	00270000
43	0	LOC	3	00020003
44	0	LOC	0	00270000
45	0	MUL	1000	001403E8
46	0	STO	0	00280000
47	0	DEL	0	002A0000
48	0	XIT	68	00040044
49	0	BRS	0	002E0000
50	0	ENT	143	0003008F
51	0	LOC	2	00020002
52	0	INT	2	00050002
53	0	STO	0	00280000
54	0	DEL	0	002A0000
55	0	LOC	4	00020004
56	0	LOC	2	00020002
57	0	LOC	0	00270000
58	0	STO	0	00280000
59	0	DEL	0	002A0000
60	0	LOC	6	00020006
61	0	LOC	4	00020004
62	0	LOC	0	00270000
63	0	LOC	3	00020003
64	0	LOC	0	00270000
65	0	MUL	980	00140304
66	0	STO	0	00280000
67	0	DEL	0	002A0000
68	0	ENT	153	00030099
69	0	LOC	5	00020005
70	0	LOC	2	00020002
71	0	LOC	0	00270000
72	0	STO	0	00280000
73	0	DEL	0	002A0000
74	0	LOC	8	00020008
75	0	LOC	5	00020005
76	0	LOC	0	00270000
77	0	LOC	3	00020003
78	0	LOC	0	00270000
79	0	MUL	C 992	001483E0
80	0	STO	0	00280000
81	0	DEL	0	002A0000
82	0	XIT	97	00040061
83	0	BRS	0	002E0000
84	0	ENT	133	00030085
85	0	LOC	2	00020002
86	0	INT	3	00050003
87	0	STO	0	00280000
88	0	DEL	0	002A0000
89	0	LOC	10	0002000A
90	0	LOC	3	00020003
91	0	LOC	0	00270000
92	0	LOC	2	00020002
93	0	LOC	0	00270000
94	0	MUL	976	00140300
95	0	STO	0	00280000
96	0	DEL	0	002A0000
97	0	ENT	158	0003009E
98	0	LOC	5	00020005
99	0	LOC	3	00020003
100	0	LOC	0	00270000
101	0	STO	0	00280000
102	0	DEL	0	002A0000
103	0	LOC	9	00020009
104	0	LOC	2	00020002
105	0	LOC	0	00270000
106	0	LOC	3	00020003
107	0	LOC	0	00270000
108	0	MUL	984	00140308
109	0	STO	0	00280000
110	0	DEL	0	002A0000
111	0	XIT	8	00040008
112	0	BRS	0	002E0000
113	0	ENT	123	00030078

\*\*\*\*\*

COMM SUBEXP ELIM

FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:





FINAL POCL FOR BLOCK# 1 IS:  
(NULL)

FINAL POCL FOR BLOCK# 2 IS:  
A(2,\*,143) A(1,\*,144)

FINAL POCL FOR BLOCK# 3 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 4 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 5 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 6 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 7 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 8 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 9 IS:  
A(1,\*,126) A(3,\*,127) A(2,\*,128) C(128:127,\*,129)

FINAL POCL FOR BLOCK# 10 IS:  
A(1,\*,139) A(3,\*,140) A(2,\*,141)

BASIC BLOCK #1  
BEGINNING AT 0, ENDING AT 8  
BLOCK TRAVERSED 1 TIMES  
0 REFERENCES:  
BASIC BLOCK #2  
BEGINNING AT 8, ENDING AT 16  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
1  
BASIC BLOCK #3  
BEGINNING AT 113, ENDING AT 113  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
2  
BASIC BLOCK #4  
BEGINNING AT 17, ENDING AT 25  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
2  
BASIC BLOCK #5  
BEGINNING AT 84, ENDING AT 97  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
4  
BASIC BLOCK #6  
BEGINNING AT 26, ENDING AT 34  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
4  
BASIC BLOCK #7  
BEGINNING AT 50, ENDING AT 68  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
6  
BASIC BLOCK #8  
BEGINNING AT 35, ENDING AT 49





```

BLOCK TRAVERSED 1 TIMES
1 REFERENCES:
6
BASIC BLOCK #9
BEGINNING AT 68, ENDING AT 83
BLOCK TRAVERSED 2 TIMES
1 REFERENCES:
8
BASIC BLOCK #10
BEGINNING AT 97, ENDING AT 112
BLOCK TRAVERSED 3 TIMES
1 REFERENCES:
9

```

BLOCK SUMMARY DATA:

BLOCK NUMBER	NUMBER OF PASSES
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	2
10	3

TOTAL NUMBER OF BLOCKS PROCESSED WAS 13  
 USING THE LAST-IN-FIRST-OUT BLOCK SELECTION ALGORITHM.  
 TIME FOR THE OPTIMIZATION WAS 0.333 SECONDS.

\*\*\*CODE SYNTHESIS FILTER IS COMPLETE\*\*\*



\*\*\*CODE SYNTHESIS FILTER\*\*\*

BLOCK SELECTION METHOD WILL BE STEEPEST DESCENT (MINIMUM CURRENT POOL)

\*\*\*C O M P L E T E T A B L E D U M P\*\*\*  
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 0

	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
CONSTANTS														
CONSYM	10	20	30											
CCNTYPE	INT	INT	INT											
CONVAL	114	115	116											
CONINT	10	20	30											

ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

VALUE STACK (TOP AT 127)

	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
CONT STK														
BLOCK#	0													
ENTRY	0													

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

\*\*\*\* FORMATTED INTERMEDIATE CODE DUMP \*\*\*\*

LOC	OFF	OP CODE	CNS	ADR	RAW CODE	OPTIMIZATION
SET			/ETC			TYPE
0	0	ENT	C	114	00038072	
1	0	TOGGLE		257	00080101	
2	0	LOC		1	00020001	
3	0	INT		0	00050000	
4	0	STD		0	00290000	
5	0	LOC		2	00020002	
6	0	INT		0	00050000	
7	0	STD		0	00290000	
8	0	ENT		1	00030001	
9	0	LOC		1	00020001	
10	0	LOC		0	00270000	
11	0	LOC		2	00020002	
12	0	LOC		0	00270000	
13	0	GTR		0	00210000	
14	0	XIT		113	00040071	
15	0	XIT		17	00040011	
16	0	BSC		0	002F0000	
17	0	ENT		1	00030001	
18	0	LOC		1	00020001	
19	0	LOC		0	00270000	
20	0	LOC		2	00020002	
21	0	LOC		0	00270000	
22	0	GTR		0	00210000	
23	0	XIT		84	00040054	
24	0	XIT		26	0004001A	
25	0	BSC		0	002F0000	
26	0	ENT		1	00030001	
27	0	LOC		1	00020001	
28	0	LOC		0	00270000	
29	0	LOC		2	00020002	
30	0	LOC		0	00270000	
31	0	GTR		0	00210000	
32	0	XIT		50	00040032	
33	0	XIT		35	00040023	
34	0	BSC		0	002F0000	
35	0	ENT		1	00030001	
36	0	LOC		2	00020002	
37	0	INT		1	00050001	
38	0	STD		0	00280000	
39	0	DEL		0	002A0000	
40	0	LOC		1	00020001	
41	0	LOC		2	00020002	



42	0	LOD	0	00270000
43	00	LOC	3	00020003
44	00	LOD	00	00270000
45	00	MUL	00	00140000
46	00	STO	00	00280000
47	00	DEL	00	002A0000
48	00	XIT	68	00040044
49	00	BRS	0	002E0000
50	00	ENT	1	00030001
51	00	LOC	2	00020002
52	00	INT	2	00050002
53	00	STO	00	00280000
54	00	DEL	00	002A0000
55	00	LOC	4	00020004
56	00	LOC	2	00020002
57	00	LOD	00	00270000
58	00	STO	00	00280000
59	00	DEL	00	002A0000
60	00	LOC	6	00020006
61	00	LOC	4	00020004
62	00	LOD	00	00270000
63	00	LOC	3	00020003
64	00	LOD	00	00270000
65	00	MUL	00	00140000
66	00	STO	00	00280000
67	00	DEL	00	002A0000
68	00	ENT	1	00030001
69	00	LOC	5	00020005
70	00	LOC	2	00020002
71	00	LOD	00	00270000
72	00	STO	00	00280000
73	00	DEL	00	002A0000
74	00	LOC	8	00020008
75	00	LOC	5	00020005
76	00	LOD	00	00270000
77	00	LOC	3	00020003
78	00	LOD	00	00270000
79	00	MUL	00	00140000
80	00	STO	00	00280000
81	00	DEL	00	002A0000
82	00	XIT	97	00040061
83	00	BRS	0	002E0000
84	00	ENT	1	00030001
85	00	LOC	2	00020002
86	00	INT	3	00050003
87	00	STO	00	00280000
88	00	DEL	00	002A0000
89	00	LOC	10	0002000A
90	00	LOC	3	00020003
91	00	LOD	00	00270000
92	00	LOC	2	00020002
93	00	LOD	00	00270000
94	00	MUL	00	00140000
95	00	STO	00	00280000
96	00	DEL	00	002A0000
97	00	ENT	1	00030001
98	00	LOC	5	00020005
99	00	LOC	3	00020003
100	00	LOD	00	00270000
101	00	STO	00	00280000
102	00	DEL	00	002A0000
103	00	LOC	9	00020009
104	00	LOC	2	00020002
105	00	LOD	00	00270000
106	00	LOC	3	00020003
107	00	LOD	00	00270000
108	00	MUL	00	00140000
109	00	STO	00	00280000
110	00	DEL	00	002A0000
111	00	XIT	8	00040008
112	00	BRS	0	002E0000
113	00	ENT	1	00030001

\*\*\*\*\*

BASIC BLOCK #1  
 BEGINNING AT 0, ENDING AT 0  
 BLOCK TRAVERSED 0 TIMES  
 0 REFERENCES:

\*\*\*END OF COMPLETE TABLE DUMP\*\*\*

B+2	0	LOC	1	00020001
B+3	0	INT	0	00050000
B+4	0	STD	0	00290000
B+5	0	LOC	2	00020002
B+6	0	INT	0	00050000
B+7	0	STD	0	00290000



B+8 0 1ENT 1 11 1 00030001 1

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 2 IS:  
A(2,\*,118) A(1,\*,117)

CONTROL AT 8, FROM BLK 1 TO BLK 2 (PASS 1)			
C+8	0	ENT	C 118 00038076
B+9	0	LOC	1 00020001
B+10	0	LOD	0 00270000
B+11	0	LOC	2 00020002
B+12	0	LOD	0 00270000
B+13	0	GTR	0 00210000
B+14	0	XIT	113 00040071
B+15	0	XIT	17 00040011
B+16	0	BSC	0 002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 3 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 4 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

CONTROL AT 17, FROM BLK 2 TO BLK 4 (PASS 1)			
C+17	0	ENT	C 128 00038080
B+18	0	LOC	1 00020001
B+19	0	LOD	0 00270000
B+20	0	LOC	2 00020002
B+21	0	LOD	0 00270000
B+22	0	GTR	0 00210000
B+23	0	XIT	84 00040054
B+24	0	XIT	26 0004001A
B+25	0	BSC	0 002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 5 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 6 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

CONTROL AT 26, FROM BLK 4 TO BLK 6 (PASS 1)			
C+26	0	ENT	C 138 0003808A
B+27	0	LOC	1 00020001
B+28	0	LOD	0 00270000
B+29	0	LOC	2 00020002
B+30	0	LOD	0 00270000
B+31	0	GTR	0 00210000
B+32	0	XIT	50 00040032
B+33	0	XIT	35 00040023
B+34	0	BSC	0 002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 7 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

INITIAL CURRENT OPTIMIZED PCOL TO BE USED FOR BLOCK# 8 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

CONTROL AT 35, FROM BLK 6 TO BLK 8 (PASS 1)			
C+35	0	ENT	C 148 00038094
B+36	0	LOC	2 00020002
B+37	0	INT	1 00050001
B+38	0	STO	0 00280000
B+39	0	DEL	0 002A0000
B+40	0	LOC	1 00020001
B+41	0	LOC	2 00020002
B+42	0	LOD	0 00270000
B+43	0	LOC	3 00020003
B+44	0	LOD	0 00270000
B+45	0	MUL	0 00140000
B+46	0	STO	0 00280000
B+47	0	DEL	0 002A0000
B+48	0	XIT	68 00040044
B+49	0	BRS	0 002E0000

INITIAL CURRENT OPTIMIZED PCOL TO BE USED FOR BLOCK# 9 IS:  
C(117:118,\*,119) C(114:120,\*,121) A(1,\*,121) A(3,\*,120) A(2,114,114)

CONTROL AT 50, FROM BLK 6 TO BLK 7 (PASS 1)			
C+50	0	ENT	C 143 0003808F
B+51	0	LOC	2 00020002
B+52	0	INT	2 00050002
B+53	0	STO	0 00280000
B+54	0	DEL	0 002A0000
B+55	0	LOC	4 00020004
B+56	0	LOC	2 00020002
B+57	0	LOD	0 00270000





B+58	0	STO	0	00280000
B+59	0	DEL	0	002A0000
B+60	0	LOC	6	00020006
B+61	0	LOC	4	00020004
B+62	0	LOC	0	00270000
B+63	0	LOC	3	00020003
B+64	0	LOC	0	00270000
B+65	0	MUL	0	00140000
B+66	0	STO	0	00280000
B+67	0	DEL	0	002A0000
B+68	0	ENT	C153	00038099

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 7 IS:

C(117:118,\*,119) C(115:122,\*,123) A(3,\*,122) A(6,\*,123) A(4,115,115)  
A(2,115,115) A(1,\*,117)

CURRENT POOL FOR BLOCK# 9 IS:

C(117:118,\*,119) C(114:120,\*,121) A(1,\*,121) A(3,\*,120) A(2,114,114)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 9 IS:

A(1,\*,124) A(3,\*,125) A(2,\*,126) C(126:125,\*,127)

CONTROL AT 84, FROM BLK 4 TO BLK 5 (PASS 1)

C+84	0	ENT	C133	00038085
B+85	0	LOC	2	00020002
B+86	0	INT	3	00050003
B+87	0	STO	0	00280000
B+88	0	DEL	0	002A0000
B+89	0	LOC	10	0002000A
B+90	0	LOC	3	00020003
B+91	0	LOC	0	00270000
B+92	0	LOC	2	00020002
B+93	0	LOC	0	00270000
B+94	0	MUL	0	00140000
B+95	0	STO	0	00280000
B+96	0	DEL	0	002A0000
B+97	0	ENT	1	00030001

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 10 IS:

C(117:118,\*,119) C(128:116,\*,129) A(3,\*,128) A(10,\*,129) A(2,116,116)  
A(1,\*,117)

CONTROL AT 113, FROM BLK 2 TO BLK 3 (PASS 1)

C+113	0	ENT	C123	00038078
CONTROL AT 68, FROM BLK 7 TO BLK 9 (PASS 1)				
C+68	0	ENT	C153	00038099
B+69	0	LOC	5	00020005
B+70	0	LOC	2	00020002
B+71	0	LOC	0	00270000
B+72	0	STO	0	00280000
B+73	0	DEL	0	002A0000
B+74	0	LOC	8	00020008
B+75	0	LOC	5	00020005
B+76	0	LOC	0	00270000
B+77	0	LOC	3	00020003
B+78	0	LOC	0	00270000
B+79	0	MUL	0	00140000
B+80	0	STO	0	00280000
B+81	0	DEL	0	002A0000
B+82	0	XIT	97	00040061
B+83	0	BRS	0	002E0000

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 9 IS:

C(126:125,\*,127) A(8,\*,127) A(5,\*,126) A(1,\*,124) A(3,\*,125) A(2,\*,126)

CURRENT POOL FOR BLOCK# 10 IS:

C(117:118,\*,119) C(128:116,\*,129) A(3,\*,128) A(10,\*,129) A(2,116,116)  
A(1,\*,117)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 10 IS:

A(3,\*,130) A(2,\*,131) A(1,\*,132) C(131:130,\*,133)

CONTROL AT 97, FROM BLK 9 TO BLK 10 (PASS 1)

C+97	0	ENT	C158	0003809E
B+98	0	LOC	5	00020005
B+99	0	LOC	3	00020003



B+100	0	LOU	0	00270000
B+101	0	STO	0	00280000
B+102	0	DEL	0	002A0000
B+103	0	LOC	9	00020009
B+104	0	LOC	2	00020002
B+105	0	LOD	0	00270000
B+106	0	LOC	3	00020003
B+107	0	LOD	0	00270000
B+108	0	MUL	0	00140000
B+109	0	STO	0	00280000
B+110	0	DEL	0	002A0000
B+111	0	XIT	8	00040008
B+112	0	BRS	0	002E0000

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 10 IS:

C(131:130,\*,133) A(9,\*,133) A(5,\*,130) A(3,\*,130) A(2,\*,131) A(1,\*,132)

CURRENT POOL FOR BLOCK# 2 IS:

A(2,\*,118) A(1,\*,117)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

### FINAL OPTIMIZATION RESULTS

#####

OPTIMIZATION AT 22  
(117,118|13,119) COMM SUBEXP ELIM

OPTIMIZATION AT 31  
(117,118|13,119) COMM SUBEXP ELIM

OPTIMIZATION AT 79  
(126,125|79,127) COMM SUBEXP ELIM

OPTIMIZATION AT 108  
(130,131|108,133) COMM SUBEXP ELIM

\*\*\*\* FORMATTED INTERMEDIATE CODE DUMP \*\*\*\*

LOC	OFF	OP CODE	CNS	ADR	RAW CODE	OPTIMIZATION TYPE
SET			/ETC			
0	0	ENT		114	00030072	
1	0	TOGGLE		257	00080101	
2	0	LOC		1	00020001	
3	0	INT		0	00050000	
4	0	STD		0	00290000	
5	0	LOC		2	00020002	
6	0	INT		0	00050000	
7	0	STD		0	00290000	
8	0	ENT		118	00030076	
9	0	LOC		1	00020001	
10	0	LOD		0	00270000	
11	0	LOC		2	00020002	
12	0	LOD		0	00270000	
13	0	GTR		1024	00210400	
14	0	XIT		113	00040071	
15	0	XIT		17	00040011	
16	0	BSC		0	002F0000	
17	0	ENT		128	00030080	
18	0	LOC		1	00020001	
19	0	LOD		0	00270000	
20	0	LOC		2	00020002	
21	0	LOD		0	00270000	
22	0	GTR	C	1016	002183F8	COMM SUBEXP ELIM
23	0	XIT		84	00040054	
24	0	XIT		26	0004001A	
25	0	BSC		0	002F0000	
26	0	ENT		138	0003008A	
27	0	LOC		1	00020001	
28	0	LOD		0	00270000	
29	0	LOC		2	00020002	
30	0	LOD		0	00270000	
31	0	GTR	C	1008	002183F0	COMM SUBEXP ELIM
32	0	XIT		50	00040032	
33	0	XIT		35	00040023	
34	0	BSC		0	002F0000	
35	0	ENT		148	00030094	
36	0	LOC		2	00020002	
37	0	INT		1	00050001	



38	0	STU	0	00280000	
39	0	DEL	0	002A0000	
40	0	LOC	1	00020001	
41	0	LOC	2	00020002	
42	0	LOC	0	00270000	
43	0	LOC	3	00020003	
44	0	LOC	0	00270000	
45	0	MUL	1000	001403E8	
46	0	STO	0	00280000	
47	0	DEL	0	002A0000	
48	0	XIT	68	00040044	
49	0	BRS	0	002E0000	
50	0	ENT	143	0003008F	
51	0	LOC	2	00020002	
52	0	INT	2	00050002	
53	0	STO	0	00280000	
54	0	DEL	0	002A0000	
55	0	LOC	4	00020004	
56	0	LOC	2	00020002	
57	0	LOC	0	00270000	
58	0	STO	0	00280000	
59	0	DEL	0	002A0000	
60	0	LOC	6	00020006	
61	0	LOC	4	00020004	
62	0	LOC	0	00270000	
63	0	LOC	3	00020003	
64	0	LOC	0	00270000	
65	0	MUL	992	001403E0	
66	0	STO	0	00280000	
67	0	DEL	0	002A0000	
68	0	ENT	153	00030099	
69	0	LOC	5	00020005	
70	0	LOC	2	00020002	
71	0	LOC	0	00270000	
72	0	STO	0	00280000	
73	0	DEL	0	002A0000	
74	0	LOC	8	00020008	
75	0	LOC	5	00020005	
76	0	LOC	0	00270000	
77	0	LOC	3	00020003	
78	0	LOC	0	00270000	
79	0	MUL	C 980	001483D4	COMM SUBEXP ELIM
80	0	STO	0	00280000	
81	0	DEL	0	002A0000	
82	0	XIT	97	00040051	
83	0	BRS	0	002E0000	
84	0	ENT	133	00030085	
85	0	LOC	2	00020002	
86	0	INT	3	00050003	
87	0	STO	0	00280000	
88	0	DEL	0	002A0000	
89	0	LOC	10	0002000A	
90	0	LOC	3	00020003	
91	0	LOC	0	00270000	
92	0	LOC	2	00020002	
93	0	LOC	0	00270000	
94	0	MUL	988	001403DC	
95	0	STO	0	00280000	
96	0	DEL	0	002A0000	
97	0	ENT	158	0003009E	
98	0	LOC	5	00020005	
99	0	LOC	3	00020003	
100	0	LOC	0	00270000	
101	0	STO	0	00280000	
102	0	DEL	0	002A0000	
103	0	LOC	9	00020009	
104	0	LOC	2	00020002	
105	0	LOC	0	00270000	
106	0	LOC	3	00020003	
107	0	LOC	0	00270000	
108	0	MUL	C 976	001483D0	COMM SUBEXP ELIM
109	0	STO	0	00280000	
110	0	DEL	0	002A0000	
111	0	XIT	8	00040008	
112	0	BRS	0	002E0000	
113	0	ENT	123	0003007B	

\*\*\*\*\*

FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

FINAL POOL FOR BLOCK# 1 IS:  
(NULL)





FINAL POCL FOR BLOCK# 2 IS:  
A(2,\*,134) A(1,\*,135)

FINAL POCL FOR BLOCK# 3 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 4 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 5 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 6 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 7 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 8 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 9 IS:  
A(1,\*,124) A(3,\*,125) A(2,\*,126) C(126:125,\*,127)

FINAL POCL FOR BLOCK# 10 IS:  
A(3,\*,130) A(2,\*,131) A(1,\*,132) C(131:130,\*,133)

BASIC BLOCK #1  
BEGINNING AT 0, ENDING AT 8  
BLOCK TRAVERSED 1 TIMES  
0 REFERENCES:  
BASIC BLOCK #2  
BEGINNING AT 8, ENDING AT 16  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
1  
BASIC BLOCK #3  
BEGINNING AT 113, ENDING AT 113  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
2  
BASIC BLOCK #4  
BEGINNING AT 17, ENDING AT 25  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
2  
BASIC BLOCK #5  
BEGINNING AT 84, ENDING AT 97  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
4  
BASIC BLOCK #6  
BEGINNING AT 26, ENDING AT 34  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
4  
BASIC BLOCK #7  
BEGINNING AT 50, ENDING AT 68  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
6  
BASIC BLOCK #8  
BEGINNING AT 35, ENDING AT 49  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
6  
BASIC BLOCK #9  
BEGINNING AT 68, ENDING AT 83





BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:

7  
BASIC BLOCK #10  
BEGINNING AT 97, ENDING AT 112  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
9

BLOCK SUMMARY DATA:

BLOCK NUMBER	NUMBER OF PASSES
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1

TOTAL NUMBER OF BLOCKS PROCESSED WAS 10  
USING THE STEEPEST DESCENT (MINIMUM CURRENT POOL) BLOCK SELECTION ALGORITHM.  
TIME FOR THE OPTIMIZATION WAS 0.267 SECONDS.

\*\*\*CODE SYNTHESIS FILTER IS COMPLETE\*\*\*



\*\*\*CODE SYNTHESIS FILTER\*\*\*

BLOCK SELECTION METHOD WILL BE FIRST-IN-FIRST-OUT

\*\*\*C O M P L E T E T A B L E D U M P\*\*\*  
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 0

	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
CONSTANTS														
CONSYM	10	20	30											
CONTYPE	INT	INT	INT											
CONVAL	114	115	116											
CONINT	10	20	30											

	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
ADDR TABL														
(EMPTY)														

VALUE STACK (TOP AT 127)

	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
CONT STK														
BLOCK#	0													
ENTRY	0													

	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
EXEC STAC														
(EMPTY)														

\*\*\*\* FORMATTED INTERMEDIATE CODE DUMP \*\*\*\*

LOC	OFF	OP CODE	CNS ADR	RAW CODE	OPTIMIZATION
SET			/ETC		TYPE
0	0	ENT	C 114	00038072	
1	0	TOGGLE	257	00080101	
2	0	LOC	1	00020001	
3	0	INT	0	00050000	
4	0	STD	0	00290000	
5	0	LOC	2	00020002	
6	0	INT	0	00050000	
7	0	STD	0	00290000	
8	0	ENT	1	00030001	
9	0	LOC	1	00020001	
10	0	LOD	0	00270000	
11	0	LOC	2	00020002	
12	0	LOD	0	00270000	
13	0	GTR	0	00210000	
14	0	XIT	113	00040071	
15	0	XIT	17	00040011	
16	0	BSC	0	002F0000	
17	0	ENT	1	00030001	
18	0	LOC	1	00020001	
19	0	LOD	0	00270000	
20	0	LOC	2	00020002	
21	0	LOD	0	00270000	
22	0	GTR	0	00210000	
23	0	XIT	84	00040054	
24	0	XIT	26	0004001A	
25	0	BSC	0	002F0000	
26	0	ENT	1	00030001	
27	0	LOC	1	00020001	
28	0	LOD	0	00270000	
29	0	LOC	2	00020002	
30	0	LOD	0	00270000	
31	0	GTR	0	00210000	
32	0	XIT	50	00040032	
33	0	XIT	35	00040023	
34	0	BSC	0	002F0000	
35	0	ENT	1	00030001	
36	0	LOC	2	00020002	
37	0	INT	1	00050001	
38	0	STO	0	00280000	
39	0	DEL	0	002A0000	
40	0	LOC	1	00020001	
41	0	LOC	2	00020002	



42	0	LOC	0	00270000
43	0	LOC	3	00020003
44	0	LOC	0	00270000
45	0	MUL	0	00140000
46	0	STO	0	00280000
47	0	DEL	0	002A0000
48	0	XIT	68	00040044
49	0	BRS	0	002E0000
50	0	ENT	1	00030001
51	0	LOC	2	00020002
52	0	INT	2	00050002
53	0	STO	0	00280000
54	0	DEL	0	002A0000
55	0	LOC	4	00020004
56	0	LOC	2	00020002
57	0	LOC	0	00270000
58	0	STO	0	00280000
59	0	DEL	0	002A0000
60	0	LOC	6	00020006
61	0	LOC	4	00020004
62	0	LOC	0	00270000
63	0	LOC	3	00020003
64	0	LOC	0	00270000
65	0	MUL	0	00140000
66	0	STO	0	00280000
67	0	DEL	0	002A0000
68	0	ENT	1	00030001
69	0	LOC	5	00020005
70	0	LOC	2	00020002
71	0	LOC	0	00270000
72	0	STO	0	00280000
73	0	DEL	0	002A0000
74	0	LOC	8	00020008
75	0	LOC	5	00020005
76	0	LOC	0	00270000
77	0	LOC	3	00020003
78	0	LOC	0	00270000
79	0	MUL	0	00140000
80	0	STO	0	00280000
81	0	DEL	0	002A0000
82	0	XIT	97	00040061
83	0	BRS	0	002E0000
84	0	ENT	1	00030001
85	0	LOC	2	00020002
86	0	INT	3	00050003
87	0	STO	0	00280000
88	0	DEL	0	002A0000
89	0	LOC	10	0002000A
90	0	LOC	3	00020003
91	0	LOC	0	00270000
92	0	LOC	2	00020002
93	0	LOC	0	00270000
94	0	MUL	0	00140000
95	0	STO	0	00280000
96	0	DEL	0	002A0000
97	0	ENT	1	00030001
98	0	LOC	5	00020005
99	0	LOC	3	00020003
100	0	LOC	0	00270000
101	0	STO	0	00280000
102	0	DEL	0	002A0000
103	0	LOC	9	00020009
104	0	LOC	2	00020002
105	0	LOC	0	00270000
106	0	LOC	3	00020003
107	0	LOC	0	00270000
108	0	MUL	0	00140000
109	0	STO	0	00280000
110	0	DEL	0	002A0000
111	0	XIT	8	00040008
112	0	BRS	0	002E0000
113	0	ENT	1	00030001

\*\*\*\*\*

BASIC BLOCK #1  
 BEGINNING AT 0, ENDING AT 0  
 BLOCK TRAVERSED 0 TIMES  
 0 REFERENCES:

\*\*\*END OF COMPLETE TABLE DUMP\*\*\*

B+2	0	LOC	1	00020001
B+3	0	INT	0	00050000
B+4	0	STO	0	00290000
B+5	0	LOC	2	00020002
B+6	0	INT	0	00050000
B+7	0	STO	0	00290000



5+8 0 1ENT 1 11 1 00030001 1

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 2 IS:  
A(2,\*,118) A(1,\*,117)

CONTROL AT 8, FROM BLK 1 TO BLK 2 (PASS 1)			
C+8	0	ENT	C118 00038076
B+9	0	LOC	1 00020001
B+10	0	LOD	0 00270000
B+11	0	LCC	2 00020002
B+12	0	LOD	0 00270000
B+13	0	GTR	0 00210000
B+14	0	XIT	113 00040071
B+15	0	XIT	17 00040011
B+16	0	BSC	0 002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 3 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 4 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

CONTROL AT 113, FROM BLK 2 TO BLK 3 (PASS 1)			
C+113	0	ENT	C1123 00038078

CONTROL AT 17, FROM BLK 2 TO BLK 4 (PASS 1)			
C+17	0	ENT	C128 00038080
B+18	0	LOC	1 00020001
B+19	0	LOD	0 00270000
B+20	0	LCC	2 00020002
B+21	0	LOD	0 00270000
B+22	0	GTR	0 00210000
B+23	0	XIT	84 00040054
B+24	0	XIT	26 0004001A
B+25	0	BSC	0 002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 5 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 6 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

CONTROL AT 84, FROM BLK 4 TO BLK 5 (PASS 1)			
C+84	0	ENT	C1133 00038085
B+85	0	LOC	2 00020002
B+86	0	INT	3 00050003
B+87	0	STO	0 00280000
B+88	0	DEL	0 002A0000
B+89	0	LOC	10 0002000A
B+90	0	LOC	3 00020003
B+91	0	LOD	0 00270000
B+92	0	LOC	2 00020002
B+93	0	LOD	0 00270000
B+94	0	MUL	0 00140000
B+95	0	STO	0 00280000
B+96	0	DEL	0 002A0000
B+97	0	ENT	1 00030001

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 7 IS:  
C(117:118,\*,119) C(120:116,\*,121) A(3,\*,120) A(10,\*,121) A(2,116,116)  
A(1,\*,117)

CONTROL AT 26, FROM BLK 4 TO BLK 6 (PASS 1)			
C+26	0	ENT	C1138 0003808A
B+27	0	LOC	1 00020001
B+28	0	LOD	0 00270000
B+29	0	LOC	2 00020002
B+30	0	LOD	0 00270000
B+31	0	GTR	0 00210000
B+32	0	XIT	50 00040032
B+33	0	XIT	35 00040023
B+34	0	BSC	0 002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 8 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 9 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

CONTROL AT 97, FROM BLK 5 TO BLK 7 (PASS 1)			
C+97	0	ENT	C1143 0003808F
B+98	0	LOC	5 00020005
B+99	0	LOC	3 00020003
B+100	0	LOD	0 00270000
B+101	0	STO	0 00280000
B+102	0	DEL	0 002A0000





B+103	0	LOC	2	00020009
B+104	0	LOC	2	00020002
B+105	0	LOC	0	00270000
B+106	0	LOC	3	00020003
B+107	0	LOC	0	00270000
B+108	0	MUL	0	00140000
B+109	0	STO	0	00280000
B+110	0	DEL	0	002A0000
B+111	0	XIT	8	00040008
B+112	0	BRS	0	002E0000

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 7 IS:

C(120:116,\*,121) C(117:118,\*,119) A(9,\*,121) A(5,\*,120) A(3,\*,120)  
A(10,\*,121) A(2,116,116) A(1,\*,117)

CURRENT POOL FOR BLOCK# 2 IS:

A(2,\*,118) A(1,\*,117)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

CONTROL AT 50, FROM BLK 6 TO BLK 8 (PASS 1)

C+50	0	ENT	C 148	00038094
B+51	0	LOC	2	00020002
B+52	0	INT	2	00050002
B+53	0	STO	0	00280000
B+54	0	DEL	0	002A0000
B+55	0	LOC	4	00020004
B+56	0	LOC	2	00020002
B+57	0	LOC	0	00270000
B+58	0	STO	0	00280000
B+59	0	DEL	0	002A0000
B+60	0	LOC	6	00020006
B+61	0	LOC	4	00020004
B+62	0	LOC	0	00270000
B+63	0	LOC	3	00020003
B+64	0	LOC	0	00270000
B+65	0	MUL	0	00140000
B+66	0	STO	0	00280000
B+67	0	DEL	0	002A0000
B+68	0	ENT	1	00030001

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 10 IS:

C(117:118,\*,119) C(115:123,\*,124) A(3,\*,123) A(6,\*,124) A(4,115,115)  
A(2,115,115) A(1,\*,117)

CONTROL AT 35, FROM BLK 6 TO BLK 9 (PASS 1)

C+35	0	ENT	C 153	00038099
B+36	0	LOC	2	00020002
B+37	0	INT	1	00050001
B+38	0	STO	0	00280000
B+39	0	DEL	0	002A0000
B+40	0	LOC	1	00020001
B+41	0	LOC	2	00020002
B+42	0	LOC	0	00270000
B+43	0	LOC	3	00020003
B+44	0	LOC	0	00270000
B+45	0	MUL	0	00140000
B+46	0	STO	0	00280000
B+47	0	DEL	0	002A0000
B+48	0	XIT	68	00040044
B+49	0	BRS	0	002E0000

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 9 IS:

C(117:118,\*,119) C(114:125,\*,126) A(1,\*,126) A(3,\*,125) A(2,114,114)

CURRENT POOL FOR BLOCK# 10 IS:

C(117:118,\*,119) C(115:123,\*,124) A(3,\*,123) A(6,\*,124) A(4,115,115)  
A(2,115,115) A(1,\*,117)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 10 IS:

A(3,\*,127) A(2,\*,128) A(1,\*,129)

CONTROL AT 68, FROM BLK 9 TO BLK 10 (PASS 1)

C+68	0	ENT	C 158	0003809E
B+69	0	LOC	5	00020005
B+70	0	LOC	2	00020002
B+71	0	LOC	0	00270000
B+72	0	STO	0	00280000



B+73	U	DEL	U	002A0000
B+74	0	LOC	8	00020008
B+75	0	LOC	5	00020005
B+76	0	LOC	0	00270000
B+77	0	LOC	3	00020003
B+78	0	LOC	0	00270000
B+79	0	MUL	0	00140000
B+80	0	STD	0	00280000
B+81	0	DEL	0	002A0000
B+82	0	XIT	97	00040061
B+83	0	BRS	0	002E0000

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 10 IS:  
C(128:127,\*,130) A(8,\*,130) A(5,\*,128) A(3,\*,127) A(2,\*,128) A(1,\*,129)

CURRENT POOL FOR BLOCK# 7 IS:  
C(117:118,\*,119) C(120:116,\*,121) A(3,\*,120) A(10,\*,121) A(2,116,116)  
A(1,\*,117)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 7 IS:  
A(3,\*,131) A(2,\*,132) A(1,\*,133) C(132:131,\*,134)

CONTROL AT 97, FROM BLK 10 TO BLK 7 (PASS 2)				
C+97	0	ENT	C 143	0003808F
B+98	0	LOC	5	00020005
B+99	0	LOC	3	00020003
B+100	0	LOC	0	00270000
B+101	0	STD	0	00280000
B+102	0	DEL	0	002A0000
B+103	0	LOC	9	00020009
B+104	0	LOC	2	00020002
B+105	0	LOC	0	00270000
B+106	0	LOC	3	00020003
B+107	0	LOC	0	00270000
B+108	0	MUL	C 992	001483E0
B+109	0	STD	0	00280000
B+110	0	DEL	0	002A0000
B+111	0	XIT	8	00040008
B+112	0	BRS	0	002E0000

COMM SUBEXP ELIM

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 7 IS:  
C(132:131,\*,134) A(9,\*,134) A(5,\*,131) A(3,\*,131) A(2,\*,132) A(1,\*,133)

CURRENT POOL FOR BLOCK# 2 IS:  
A(2,\*,122) A(1,\*,117)

AFTER PERFORMING THE MEET OPERATION:  
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

### FINAL OPTIMIZATION RESULTS

#####

OPTIMIZATION AT 22  
(117,118|13,119) COMM SUBEXP ELIM

OPTIMIZATION AT 31  
(117,118|13,119) COMM SUBEXP ELIM

OPTIMIZATION AT 108  
(132,131|94,134) COMM SUBEXP ELIM

\*\*\*\* FORMATTED INTERMEDIATE CODE DUMP \*\*\*\*

LOC	OFF	OP CODE	CNS ADR	RAW CODE	OPTIMIZATION TYPE
SET			/ETC		
0	0	ENT	114	00030072	
1	0	TOGGLE	257	00080101	
2	0	LOC	1	00020001	
3	0	INT	0	00050000	
4	0	STD	0	00290000	
5	0	LOC	2	00020002	
6	0	INT	0	00050000	
7	0	STD	0	00290000	
8	0	ENT	118	00030076	



9	0	LUC	1	00020001	
10	0	LOD	0	00270000	
11	0	LOC	2	00020002	
12	0	LOD	0	00270000	
13	0	GTR	1024	00210400	
14	0	XIT	113	00040071	
15	0	XIT	17	00040011	
16	0	BSC	0	002F0000	
17	0	ENT	128	00030080	
18	0	LOC	1	00020001	
19	0	LOD	0	00270000	
20	0	LOC	2	00020002	
21	0	LOD	0	00270000	
22	0	GTR	C 1016	002183F8	COMM SUBEXP ELIM
23	0	XIT	84	00040054	
24	0	XIT	26	0004001A	
25	0	BSC	0	002F0000	
26	0	ENT	138	0003008A	
27	0	LOC	1	00020001	
28	0	LOD	0	00270000	
29	0	LOC	2	00020002	
30	0	LOD	0	00270000	
31	0	GTR	C 1000	002183E8	COMM SUBEXP ELIM
32	0	XIT	50	00040032	
33	0	XIT	35	00040023	
34	0	BSC	0	002F0000	
35	0	ENT	153	00030099	
36	0	LOC	2	00020002	
37	0	INT	1	00050001	
38	0	STO	0	00280000	
39	0	DEL	0	002A0000	
40	0	LOC	1	00020001	
41	0	LOC	2	00020002	
42	0	LOD	0	00270000	
43	0	LOC	3	00020003	
44	0	LOD	0	00270000	
45	0	MUL	980	001403D4	
46	0	STO	0	00280000	
47	0	DEL	0	002A0000	
48	0	XIT	68	00040044	
49	0	BRS	0	002E0000	
50	0	ENT	148	00030094	
51	0	LOC	2	00020002	
52	0	INT	2	00050002	
53	0	STO	0	00280000	
54	0	DEL	0	002A0000	
55	0	LOC	4	00020004	
56	0	LOC	2	00020002	
57	0	LOD	0	00270000	
58	0	STO	0	00280000	
59	0	DEL	0	002A0000	
60	0	LOC	6	00020006	
61	0	LOC	4	00020004	
62	0	LOD	0	00270000	
63	0	LOC	3	00020003	
64	0	LOD	0	00270000	
65	0	MUL	988	001403DC	
66	0	STO	0	00280000	
67	0	DEL	0	002A0000	
68	0	ENT	158	0003009E	
69	0	LOC	5	00020005	
70	0	LOC	2	00020002	
71	0	LOD	0	00270000	
72	0	STO	0	00280000	
73	0	DEL	0	002A0000	
74	0	LOC	8	00020008	
75	0	LOC	5	00020005	
76	0	LOD	0	00270000	
77	0	LOC	3	00020003	
78	0	LOD	0	00270000	
79	0	MUL	976	001403D0	
80	0	STO	0	00280000	
81	0	DEL	0	002A0000	
82	0	XIT	97	00040061	
83	0	BRS	0	002E0000	
84	0	ENT	133	00030085	
85	0	LOC	2	00020002	
86	0	INT	3	00050003	
87	0	STO	0	00280000	
88	0	DEL	0	002A0000	
89	0	LOC	10	0002000A	
90	0	LOC	3	00020003	
91	0	LOD	0	00270000	
92	0	LOC	2	00020002	
93	0	LOD	0	00270000	
94	0	MUL	1008	001403F0	
95	0	STO	0	00280000	
96	0	DEL	0	002A0000	





97	0	ENT	143	0003008F
98	0	LOC	5	00020005
99	0	LOC	3	00020003
100	0	LOD	0	00270000
101	0	STO	0	00280000
102	0	DEL	0	002A0000
103	0	LOC	9	00020009
104	0	LOC	2	00020002
105	0	LOD	0	00270000
106	0	LOC	3	00020003
107	0	LOD	0	00270000
108	0	MUL	C 992	001483E0
109	0	STO	0	00280000
110	0	DEL	0	002A0000
111	0	XIT	8	00040008
112	0	BRS	0	002E0000
113	0	ENT	123	0003007B

\*\*\*\*\*

COMM SUBEXP ELIM

FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

FINAL POCL FOR BLOCK# 1 IS:  
(NULL)

FINAL POCL FOR BLOCK# 2 IS:  
A(2,\*,135) A(1,\*,136)

FINAL POCL FOR BLOCK# 3 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 4 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 5 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 6 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 7 IS:  
A(3,\*,131) A(2,\*,132) A(1,\*,133) C(132:131,\*,134)

FINAL POCL FOR BLOCK# 8 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 9 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 10 IS:  
A(3,\*,127) A(2,\*,128) A(1,\*,129)

BASIC BLOCK #1  
BEGINNING AT 0, ENDING AT 8  
BLOCK TRAVERSED 1 TIMES  
0 REFERENCES:

BASIC BLOCK #2  
BEGINNING AT 8, ENDING AT 16  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:

BASIC BLOCK #3  
BEGINNING AT 113, ENDING AT 113  
BLOCK TRAVERSED 1 TIMES





```

1 REFERENCES:
2
BASIC BLOCK #4
BEGINNING AT 17, ENDING AT 25
BLOCK TRAVERSED 1 TIMES
1 REFERENCES:
2
BASIC BLOCK #5
BEGINNING AT 84, ENDING AT 97
BLOCK TRAVERSED 1 TIMES
1 REFERENCES:
4
BASIC BLOCK #6
BEGINNING AT 26, ENDING AT 34
BLOCK TRAVERSED 1 TIMES
1 REFERENCES:
4
BASIC BLOCK #7
BEGINNING AT 97, ENDING AT 112
BLOCK TRAVERSED 2 TIMES
1 REFERENCES:
5
BASIC BLOCK #8
BEGINNING AT 50, ENDING AT 68
BLOCK TRAVERSED 1 TIMES
1 REFERENCES:
6
BASIC BLOCK #9
BEGINNING AT 35, ENDING AT 49
BLOCK TRAVERSED 1 TIMES
1 REFERENCES:
6
BASIC BLOCK #10
BEGINNING AT 68, ENDING AT 83
BLOCK TRAVERSED 1 TIMES
1 REFERENCES:
9

```

BLOCK SUMMARY DATA:

BLOCK NUMBER	NUMBER OF PASSES
1	1
2	1
3	1
4	1
5	1
6	1
7	2
8	1
9	1
10	1

TOTAL NUMBER OF BLOCKS PROCESSED WAS 11  
 USING THE FIRST-IN-FIRST-OUT BLOCK SELECTION ALGORITHM.  
 TIME FOR THE OPTIMIZATION WAS 0.216 SECONDS.

\*\*\*CODE SYNTHESIS FILTER IS COMPLETE\*\*\*



\*\*\*CODE SYNTHESIS FILTER\*\*\*

BLOCK SELECTION METHOD WILL BE MAXIMUM CURRENT POOL

\*\*\*C O M P L E T E T A B L E D U M P\*\*\*  
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 0

	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
CONSTANTS														
CONSYM	10	20	30											
CONTYPE	INT	INT	INT											
CONVAL	114	115	116											
CONINT	10	20	30											

ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

VALUE STACK (TOP AT 127)

	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
CONT STK														
BLOCK#	0													
ENTRY	0													

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

\*\*\*\* FORMATTED INTERMEDIATE CODE DUMP \*\*\*\*

LOC	OFF	OP CODE	CNS ADR	RAW CODE	OPTIMIZATION TYPE
0	0	ENT	C 114	00038072	
1	0	TOGGLE	257	000B0101	
2	0	LOC	1	00020001	
3	0	INT	0	00050000	
4	0	STO	0	00290000	
5	0	LOC	2	00020002	
6	0	INT	0	00050000	
7	0	STO	0	00290000	
8	0	ENT	1	00030001	
9	0	LOC	1	00020001	
10	0	LOD	0	00270000	
11	0	LOC	2	00020002	
12	0	LOD	0	00270000	
13	0	GTR	0	00210000	
14	0	XIT	113	00040071	
15	0	XIT	17	00040011	
16	0	BSC	0	002F0000	
17	0	ENT	1	00030001	
18	0	LOC	1	00020001	
19	0	LOD	0	00270000	
20	0	LOC	2	00020002	
21	0	LOD	0	00270000	
22	0	GTR	0	00210000	
23	0	XIT	84	00040054	
24	0	XIT	26	0004001A	
25	0	BSC	0	002F0000	
26	0	ENT	1	00030001	
27	0	LOC	1	00020001	
28	0	LOD	0	00270000	
29	0	LOC	2	00020002	
30	0	LOD	0	00270000	
31	0	GTR	0	00210000	
32	0	XIT	50	00040032	
33	0	XIT	35	00040023	
34	0	BSC	0	002F0000	
35	0	ENT	1	00030001	
36	0	LOC	2	00020002	
37	0	INT	1	00050001	
38	0	STO	0	00280000	
39	0	DEL	0	002A0000	
40	0	LOC	1	00020001	
41	0	LOC	2	00020002	



42	0	LOD	0	00270000
43	0	LOC	3	00020003
44	0	LOD	0	00270000
45	0	MUL	0	00140000
46	0	STO	0	00280000
47	0	DEL	0	002A0000
48	0	XIT	68	00040044
49	0	BRS	0	002E0000
50	0	ENT	1	00030001
51	0	LOC	2	00020002
52	0	INT	2	00050002
53	0	STO	0	00280000
54	0	DEL	0	002A0000
55	0	LOC	4	00020004
56	0	LOC	2	00020002
57	0	LOD	0	00270000
58	0	STO	0	00280000
59	0	DEL	0	002A0000
60	0	LOC	6	00020006
61	0	LOC	4	00020004
62	0	LOD	0	00270000
63	0	LOC	3	00020003
64	0	LOD	0	00270000
65	0	MUL	0	00140000
66	0	STO	0	00280000
67	0	DEL	0	002A0000
68	0	ENT	1	00030001
69	0	LOC	5	00020005
70	0	LOC	2	00020002
71	0	LOD	0	00270000
72	0	STO	0	00280000
73	0	DEL	0	002A0000
74	0	LOC	8	00020008
75	0	LOC	5	00020005
76	0	LOD	0	00270000
77	0	LOC	3	00020003
78	0	LOD	0	00270000
79	0	MUL	0	00140000
80	0	STO	0	00280000
81	0	DEL	0	002A0000
82	0	XIT	97	00040061
83	0	BRS	0	002E0000
84	0	ENT	1	00030001
85	0	LOC	2	00020002
86	0	INT	3	00050003
87	0	STO	0	00280000
88	0	DEL	0	002A0000
89	0	LOC	10	0002000A
90	0	LOC	3	00020003
91	0	LOD	0	00270000
92	0	LOC	2	00020002
93	0	LOD	0	00270000
94	0	MUL	0	00140000
95	0	STO	0	00280000
96	0	DEL	0	002A0000
97	0	ENT	1	00030001
98	0	LOC	5	00020005
99	0	LOC	3	00020003
100	0	LOD	0	00270000
101	0	STO	0	00280000
102	0	DEL	0	002A0000
103	0	LOC	9	00020009
104	0	LOC	2	00020002
105	0	LOD	0	00270000
106	0	LOC	3	00020003
107	0	LOD	0	00270000
108	0	MUL	0	00140000
109	0	STO	0	00280000
110	0	DEL	0	002A0000
111	0	XIT	8	00040008
112	0	BRS	0	002E0000
113	0	ENT	1	00030001

\*\*\*\*\*

BASIC BLOCK #1  
 BEGINNING AT 0, ENDING AT 0  
 BLOCK TRAVERSED 0 TIMES  
 0 REFERENCES:

\*\*\*END OF COMPLETE TABLE DUMP\*\*\*

B+2	0	LOC	1	00020001
B+3	0	INT	0	00050000
B+4	0	STD	0	00290000
B+5	0	LOC	2	00020002
B+6	0	INT	0	00050000
B+7	0	STD	0	00290000



B+8 0 1ENI 1 11 1 00030001 1

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 2 IS:  
A(2,\*,118) A(1,\*,117)

CONTROL AT 8, FROM BLK 1 TO BLK 2 (PASS 1)			
C+8	0	ENT	C 118 00038076
B+9	0	LOC	1 00020001
B+10	0	LOD	0 00270000
B+11	0	LOC	2 00020002
B+12	0	LOD	0 00270000
B+13	0	GTR	0 00210000
B+14	0	XIT	113 00040071
B+15	0	XIT	17 00040011
B+16	0	BSC	0 002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 3 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 4 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

CONTROL AT 17, FROM BLK 2 TO BLK 4 (PASS 1)			
C+17	0	ENT	C 128 00038080
B+18	0	LOC	1 00020001
B+19	0	LOD	0 00270000
B+20	0	LOC	2 00020002
B+21	0	LOD	0 00270000
B+22	0	GTR	0 00210000
B+23	0	XIT	84 00040054
B+24	0	XIT	26 0004001A
B+25	0	BSC	0 002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 5 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 6 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

CONTROL AT 26, FROM BLK 4 TO BLK 6 (PASS 1)			
C+26	0	ENT	C 138 0003808A
B+27	0	LOC	1 00020001
B+28	0	LOD	0 00270000
B+29	0	LOC	2 00020002
B+30	0	LOD	0 00270000
B+31	0	GTR	0 00210000
B+32	0	XIT	50 00040032
B+33	0	XIT	35 00040023
B+34	0	BSC	0 002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 7 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 8 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

CONTROL AT 35, FROM BLK 6 TO BLK 8 (PASS 1)			
C+35	0	ENT	C 148 00038094
B+36	0	LOC	2 00020002
B+37	0	INT	1 00050001
B+38	0	STO	0 00280000
B+39	0	DEL	0 002A0000
B+40	0	LOC	1 00020001
B+41	0	LOC	2 00020002
B+42	0	LOD	0 00270000
B+43	0	LOC	3 00020003
B+44	0	LOD	0 00270000
B+45	0	MUL	0 00140000
B+46	0	STO	0 00280000
B+47	0	DEL	0 002A0000
B+48	0	XIT	68 00040044
B+49	0	BRS	0 002E0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 9 IS:  
C(117:118,\*,119) C(114:120,\*,121) A(1,\*,121) A(3,\*,120) A(2,114,114)

CONTROL AT 68, FROM BLK 8 TO BLK 9 (PASS 1)			
C+68	0	ENT	C 153 00038099
B+69	0	LOC	5 00020005
B+70	0	LOC	2 00020002
B+71	0	LOD	0 00270000
B+72	0	STO	0 00280000
B+73	0	DEL	0 002A0000
B+74	0	LOC	8 00020008
B+75	0	LOC	5 00020005





B+76	0	LUD	0	00270000
B+77	0	LOC	3	00020003
B+78	0	LUD	0	00270000
B+79	0	MUL	0	00140000
B+80	0	STO	0	00280000
B+81	0	DEL	0	002A0000
B+82	0	XIT	97	00040061
B+83	0	BRS	0	002E0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 10 IS:  
 C(114:120,\*,121) C(117:118,\*,119) A(8,\*,121) A(5,114,114) A(1,\*,121)  
 A(3,\*,120) A(2,114,114)

CONTROL AT 97, FROM BLK 9 TO BLK 10 (PASS 1)				
C+97	0	ENT	C 158	0003809E
B+98	0	LOC	5	00020005
B+99	0	LOC	3	00020003
B+100	0	LUD	0	00270000
B+101	0	STO	0	00280000
B+102	0	DEL	0	002A0000
B+103	0	LOC	9	00020009
B+104	0	LOC	2	00020002
B+105	0	LUD	0	00270000
B+106	0	LOC	3	00020003
B+107	0	LUD	0	00270000
B+108	0	MUL	0	00140000
B+109	0	STO	0	00280000
B+110	0	DEL	0	002A0000
B+111	0	XIT	8	00040008
B+112	0	BRS	0	002E0000

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 10 IS:  
 C(117:118,\*,119) C(114:120,\*,121) A(9,\*,121) A(5,\*,120) A(8,\*,121)  
 A(1,\*,121) A(3,\*,120) A(2,114,114)

CURRENT POOL FOR BLOCK# 2 IS:  
 A(2,\*,118) A(1,\*,117)

AFTER PERFORMING THE MEET OPERATION:  
 NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

CONTROL AT 50, FROM BLK 6 TO BLK 7 (PASS 1)				
C+50	0	ENT	C 143	0003808F
B+51	0	LOC	2	00020002
B+52	0	INT	2	00050002
B+53	0	STO	0	00280000
B+54	0	DEL	0	002A0000
B+55	0	LOC	4	00020004
B+56	0	LOC	2	00020002
B+57	0	LUD	0	00270000
B+58	0	STO	0	00280000
B+59	0	DEL	0	002A0000
B+60	0	LOC	6	00020006
B+61	0	LOC	4	00020004
B+62	0	LUD	0	00270000
B+63	0	LOC	3	00020003
B+64	0	LUD	0	00270000
B+65	0	MUL	0	00140000
B+66	0	STO	0	00280000
B+67	0	DEL	0	002A0000
B+68	0	ENT	C 153	00038099

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 7 IS:  
 C(117:118,\*,119) C(115:124,\*,125) A(3,\*,124) A(6,\*,125) A(4,115,115)  
 A(2,115,115) A(1,\*,117)

CURRENT POOL FOR BLOCK# 9 IS:  
 C(117:118,\*,119) C(114:120,\*,121) A(1,\*,121) A(3,\*,120) A(2,114,114)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 9 IS:  
 A(1,\*,126) A(3,\*,127) A(2,\*,128) C(128:127,\*,129)

CONTROL AT 68, FROM BLK 7 TO BLK 9 (PASS 2)				
C+68	0	ENT	C 153	00038099
B+69	0	LOC	5	00020005
B+70	0	LOC	2	00020002
B+71	0	LUD	0	00270000
B+72	0	STO	0	00280000
B+73	0	DEL	0	002A0000



B+74	0	LOC	8	00020008	
B+75	0	LOC	5	00020005	
B+76	0	LOC	0	00270000	
B+77	0	LOC	3	00020003	
B+78	0	LOC	0	00270000	
B+79	0	MUL	C 992	001483E0	COMM SUBEXP ELIM
B+80	0	STO	0	00280000	
B+81	0	DEL	0	002A0000	
B+82	0	XIT	97	00040061	
B+83	0	BRS	0	002E0000	

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 9 IS:  
C(128:127,\*,129) A(8,\*,129) A(5,\*,128) A(1,\*,126) A(3,\*,127) A(2,\*,128)

CURRENT POOL FOR BLOCK# 10 IS:  
C(114:120,\*,121) C(117:118,\*,119) A(8,\*,121) A(5,114,114) A(1,\*,121)  
A(3,\*,120) A(2,114,114)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 10 IS:  
A(8,\*,130) A(5,\*,131) A(1,\*,132) A(3,\*,133) A(2,\*,131)

CONTROL AT 97, FROM BLK 9 TO BLK 10 (PASS 2)					
C+97	0	ENT	C 158	0003809E	
B+98	0	LOC	5	00020005	
B+99	0	LOC	3	00020003	
B+100	0	LOC	0	00270000	
B+101	0	STO	0	00280000	
B+102	0	DEL	0	002A0000	
B+103	0	LOC	9	00020009	
B+104	0	LOC	2	00020002	
B+105	0	LOC	0	00270000	
B+106	0	LOC	3	00020003	
B+107	0	LOC	0	00270000	
B+108	0	MUL	C 984	001483D8	COMM SUBEXP ELIM
B+109	0	STO	0	00280000	
B+110	0	DEL	0	002A0000	
B+111	0	XIT	8	00040008	
B+112	0	BRS	0	002E0000	

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 10 IS:  
C(131:133,\*,134) A(9,\*,134) A(5,\*,133) A(8,\*,130) A(1,\*,132) A(3,\*,133)  
A(2,\*,131)

CURRENT POOL FOR BLOCK# 2 IS:  
A(2,\*,122) A(1,\*,123)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

CONTROL AT 64, FROM BLK 4 TO BLK 5 (PASS 1)					
C+84	0	ENT	C 133	00038085	
B+85	0	LOC	2	00020002	
B+86	0	INT	3	00050003	
B+87	0	STO	0	00280000	
B+88	0	DEL	0	002A0000	
B+89	0	LOC	10	0002000A	
B+90	0	LOC	3	00020003	
B+91	0	LOC	0	00270000	
B+92	0	LOC	2	00020002	
B+93	0	LOC	0	00270000	
B+94	0	MUL	0	00140000	
B+95	0	STO	0	00280000	
B+96	0	DEL	0	002A0000	
B+97	0	ENT	C 158	0003809E	

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 5 IS:  
C(117:118,\*,119) C(137:116,\*,138) A(3,\*,137) A(10,\*,138) A(2,116,116)  
A(1,\*,117)

CURRENT POOL FOR BLOCK# 10 IS:  
A(8,\*,130) A(5,\*,131) A(1,\*,132) A(3,\*,133) A(2,\*,131)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 10 IS:  
A(1,\*,139) A(3,\*,140) A(2,\*,141)



```

CONTROL AT 97, FROM BLK 5 TO BLK 10 (PASS 3)
C+97 0 ENT C 158 0003809E
B+98 0 LOC 5 00020005
B+99 0 LOC 3 00020003
B+100 0 LOD 0 00270000
B+101 0 STD 0 00280000
B+102 0 DEL 0 002A0000
B+103 0 LOC 9 00020009
B+104 0 LOC 2 00020002
B+105 0 LOD 0 00270000
B+106 0 LOC 3 00020003
B+107 0 LOD 0 00270000
B+108 0 MUL C 984 001483D8
B+109 0 STD 0 00280000
B+110 0 DEL 0 002A0000
B+111 0 XIT 8 00040008
B+112 0 BRS 0 002E0000

```

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 10 IS:

C(141:140,\*,142) A(9,\*,142) A(5,\*,140) A(1,\*,139) A(3,\*,140) A(2,\*,141)

CURRENT POOL FOR BLOCK# 2 IS:

A(2,\*,135) A(1,\*,136)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

CONTROL AT 113, FROM BLK 2 TO BLK 3 (PASS 1)

```

C+113 0 ENT C 1123 0003307B

```

#### FINAL OPTIMIZATION RESULTS

#####

OPTIMIZATION AT 22  
(117,118|13,119) COMM SUBEXP ELIM

OPTIMIZATION AT 31  
(117,118|13,119) COMM SUBEXP ELIM

OPTIMIZATION AT 79  
(127,128|45,129) COMM SUBEXP ELIM

\*\*\*\* FORMATTED INTERMEDIATE CODE DUMP \*\*\*\*

LOC	OFF	OP CODE	CNS ADR	RAW CODE	OPTIMIZATION
SET			/ETC		TYPE
0	0	ENT	114	00030072	
1	0	TOGGLE	257	00080101	
2	0	LOC	1	00020001	
3	0	INT	0	00050000	
4	0	STD	0	00290000	
5	0	LOC	2	00020002	
6	0	INT	0	00050000	
7	0	STD	0	00290000	
8	0	ENT	118	00030076	
9	0	LOC	1	00020001	
10	0	LOD	0	00270000	
11	0	LOC	2	00020002	
12	0	LOD	0	00270000	
13	0	GTR	1024	00210400	
14	0	XIT	113	00040071	
15	0	XIT	17	00040011	
16	0	BSC	0	002F0000	
17	0	ENT	128	00030080	
18	0	LOC	1	00020001	
19	0	LOD	0	00270000	
20	0	LOC	2	00020002	
21	0	LOD	0	00270000	
22	0	GTR	C 1016	002183F8	COMM SUBEXP ELIM
23	0	XIT	84	00040054	
24	0	XIT	26	0004001A	
25	0	BSC	0	002F0000	
26	0	ENT	138	0003008A	
27	0	LOC	1	00020001	
28	0	LOD	0	00270000	
29	0	LOC	2	00020002	
30	0	LOD	0	00270000	
31	0	GTR	C 1008	002183F0	COMM SUBEXP ELIM
32	0	XIT	50	00040032	





33	0	XIT	35	00040023
34	00	BSC.	0	002F0000
35	00	ENT	148	00030094
36	00	LOC	2	00020002
37	00	INT	1	00050001
38	00	STO	0	00280000
39	00	DEL	0	002A0000
40	00	LOC	1	00020001
41	00	LOC	2	00020002
42	00	LOC	0	00270000
43	00	LOC	3	00020003
44	00	LOC	0	00270000
45	00	MUL	1000	001403E8
46	00	STO	0	00280000
47	00	DEL	0	002A0000
48	00	XIT	68	00040044
49	00	BRS	0	002E0000
50	00	ENT	143	0003008F
51	00	LOC	2	00020002
52	00	INT	2	00050002
53	00	STO	0	00280000
54	00	DEL	0	002A0000
55	00	LOC	4	00020004
56	00	LOC	2	00020002
57	00	LOC	0	00270000
58	00	STO	0	00280000
59	00	DEL	0	002A0000
60	00	LOC	6	00020006
61	00	LOC	4	00020004
62	00	LOC	0	00270000
63	00	LOC	3	00020003
64	00	LOC	0	00270000
65	00	MUL	980	001403D4
66	00	STO	0	00280000
67	00	DEL	0	002A0000
68	00	ENT	153	00030099
69	00	LOC	5	00020005
70	00	LOC	2	00020002
71	00	LOC	0	00270000
72	00	STO	0	00280000
73	00	DEL	0	002A0000
74	00	LOC	8	00020008
75	00	LOC	5	00020005
76	00	LOC	0	00270000
77	00	LOC	3	00020003
78	00	LOC	0	00270000
79	00	MUL	C 992	001483E0
80	00	STO	0	00280000
81	00	DEL	0	002A0000
82	00	XIT	97	00040061
83	00	BRS	0	002E0000
84	00	ENT	133	00030085
85	00	LOC	2	00020002
86	00	INT	3	00050003
87	00	STO	0	00280000
88	00	DEL	0	002A0000
89	00	LOC	10	0002000A
90	00	LOC	3	00020003
91	00	LOC	0	00270000
92	00	LOC	2	00020002
93	00	LOC	0	00270000
94	00	MUL	976	001403D0
95	00	STO	0	00280000
96	00	DEL	0	002A0000
97	00	ENT	158	0003009E
98	00	LOC	5	00020005
99	00	LOC	3	00020003
100	00	LOC	0	00270000
101	00	STO	0	00280000
102	00	DEL	0	002A0000
103	00	LOC	9	00020009
104	00	LOC	2	00020002
105	00	LOC	0	00270000
106	00	LOC	3	00020003
107	00	LOC	0	00270000
108	00	MUL	984	001403D8
109	00	STO	0	00280000
110	00	DEL	0	002A0000
111	00	XIT	8	00040008
112	00	BRS	0	002E0000
113	0	ENT	123	0003007B

COMM SUBEXP ELIM

\*\*\*\*\*

FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:





FINAL POCL FOR BLOCK# 1 IS:  
(NULL)

FINAL POCL FOR BLOCK# 2 IS:  
A(2,\*,143) A(1,\*,144)

FINAL POCL FOR BLOCK# 3 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 4 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 5 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 6 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 7 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 8 IS:  
C(117:118,\*,119) A(2,\*,118) A(1,\*,117)

FINAL POCL FOR BLOCK# 9 IS:  
A(1,\*,126) A(3,\*,127) A(2,\*,128) C(128:127,\*,129)

FINAL POCL FOR BLOCK# 10 IS:  
A(1,\*,139) A(3,\*,140) A(2,\*,141)

BASIC BLOCK #1  
BEGINNING AT 0, ENDING AT 8  
BLOCK TRAVERSED 1 TIMES  
0 REFERENCES:  
BASIC BLOCK #2  
BEGINNING AT 8, ENDING AT 16  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
1  
BASIC BLOCK #3  
BEGINNING AT 113, ENDING AT 113  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
2  
BASIC BLOCK #4  
BEGINNING AT 17, ENDING AT 25  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
2  
BASIC BLOCK #5  
BEGINNING AT 34, ENDING AT 97  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
4  
BASIC BLOCK #6  
BEGINNING AT 26, ENDING AT 34  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
4  
BASIC BLOCK #7  
BEGINNING AT 50, ENDING AT 68  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
6  
BASIC BLOCK #8  
BEGINNING AT 35, ENDING AT 49



BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:

6  
BASIC BLOCK #9  
BEGINNING AT 68, ENDING AT 83  
BLOCK TRAVERSED 2 TIMES  
1 REFERENCES:

8  
BASIC BLOCK #10  
BEGINNING AT 97, ENDING AT 112  
BLOCK TRAVERSED 3 TIMES  
1 REFERENCES:

BLOCK SUMMARY DATA:

BLOCK NUMBER	NUMBER OF PASSES
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	2
10	3

TOTAL NUMBER OF BLOCKS PROCESSED WAS 13  
USING THE MAXIMUM CURRENT POOL BLOCK SELECTION ALGORITHM.  
TIME FOR THE OPTIMIZATION WAS 0.268 SECONDS.

\*\*\*CODE SYNTHESIS FILTER IS COMPLETE\*\*\*



CARD	BL	SYL	COMMENT
1	0	0	TEST PROGRAM # 7 ;
\$EXECUTE			
3	0	1	BEGIN LOCAL A,B,C,D;
4	1	1	A := 1;
5	1	5	B := 2;
6	1	9	C := 3;
7	1	13	A := A * 5;
8	1	20	A := A + B;
9	1	28	B := B + C;
10	1	36	B := B + 1;
11	1	43	C := A * 3;
12	1	50	C := C + 6;
13	1	57	D := 10;
14	1	61	IF A GTR B THEN
15	1	66	BEGIN
16	1	70	D := B * C;
17	2	78	END;
18	1	79	A := C * B;
19	1	87	END
20	1	87	EOF

CODE FILE COPIED (87 WORDS)

CONSTANT TABLE COPIED (12 WORDS)

2 RECORDS WRITTEN INTO FILE 1

END OF COMPILATION FEBRUARY 12, 1975. CLOCK TIME = 20:45:50.31.

20 CARDS WERE READ.

NO ERRORS WERE DETECTED.



\*\*\*CODE SYNTHESIS FILTER\*\*\*

BLOCK SELECTION METHOD WILL BE LAST-IN-FIRST-OUT

\*\*\*C O M P L E T E T A B L E D U M P\*\*\*  
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 0

CONSTANTS	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
CONSYM	1	2	3	5	6	10								
CONTYPE	INT	INT	INT	INT	INT	INT								
CONVAL	88	89	90	91	92	93								
CONINT	1	2	3	5	6	10								

ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

VALUE STACK (TOP AT 127)

CONT STK <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
BLOCK# 0  
ENTRY 0

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

\*\*\*\* FORMATTED INTERMEDIATE CODE DUMP \*\*\*\*

LOC	OFF	OP CODE	CNS ADR	RAW CODE	OPTIMIZATION
SET			/ETC		TYPE
0	0	ENT	C 88	00038058	
1	0	TOGGLE	257	00080101	
2	0	LOC	1	00020001	
3	0	INT	1	00050001	
4	0	STO	0	00280000	
5	0	DEL	0	002A0000	
6	0	LOC	2	00020002	
7	0	INT	2	00050002	
8	0	STO	0	00280000	
9	0	DEL	0	002A0000	
10	0	LOC	3	00020003	
11	0	INT	3	00050003	
12	0	STO	0	00280000	
13	0	DEL	0	002A0000	
14	0	LOC	1	00020001	
15	0	LOC	1	00020001	
16	0	LOD	0	00270000	
17	0	INT	4	00050004	
18	0	MUL	0	00140000	
19	0	STO	0	00280000	
20	0	DEL	0	002A0000	
21	0	LOC	1	00020001	
22	0	LOC	1	00020001	
23	0	LOD	0	00270000	
24	0	LOC	2	00020002	
25	0	LOD	0	00270000	
26	0	ADD	0	00100000	
27	0	STO	0	00280000	
28	0	DEL	0	002A0000	
29	0	LOC	2	00020002	
30	0	LOC	2	00020002	
31	0	LOD	0	00270000	
32	0	LOC	3	00020003	
33	0	LOD	0	00270000	
34	0	ADD	0	00100000	
35	0	STO	0	00280000	
36	0	DEL	0	002A0000	
37	0	LOC	2	00020002	
38	0	LOC	2	00020002	
39	0	LOD	0	00270000	
40	0	INT	1	00050001	
41	0	ADD	0	00100000	





42	0	STO	0	00280000
43	0	DEL	0	002A0000
44	0	LOC	3	00020003
45	0	LOC	1	00020001
46	0	LOD	0	00270000
47	0	INT	3	00050003
48	0	MUL	0	00140000
49	0	STO	0	00280000
50	0	DEL	0	002A0000
51	0	LOC	3	00020003
52	0	LOC	3	00020003
53	0	LOD	0	00270000
54	0	INT	5	00050005
55	0	ADD	0	00100000
56	0	STO	0	00280000
57	0	DEL	0	002A0000
58	0	LOC	4	00020004
59	0	INT	6	00050006
60	0	STO	0	00280000
61	0	DEL	0	002A0000
62	0	LOC	1	00020001
63	0	LOD	0	00270000
64	0	LOC	2	00020002
65	0	LOD	0	00270000
66	0	GTR	0	00210000
67	0	XIT	79	0004004F
68	0	XIT	70	00040046
69	0	BSC	0	002F0000
70	0	ENT	1	00030001
71	0	LOC	4	00020004
72	0	LOC	2	00020002
73	0	LOD	0	00270000
74	0	LOC	3	00020003
75	0	LOD	0	00270000
76	0	MUL	0	00140000
77	0	STO	0	00280000
78	0	DEL	0	002A0000
79	0	ENT	1	00030001
80	0	LOC	1	00020001
81	0	LOC	3	00020003
82	0	LOD	0	00270000
83	0	LOC	2	00020002
84	0	LOD	0	00270000
85	0	MUL	0	00140000
86	0	STO	0	00280000
87	0	DEL	0	002A0000

\*\*\*\*\*

BASIC BLOCK #1  
 BEGINNING AT 0, ENDING AT 0  
 BLOCK TRAVERSED 0 TIMES  
 0 REFERENCES:

\*\*\*END OF COMPLETE TABLE DUMP\*\*\*

B+2	0	LOC	1	00020001
B+3	0	INT	1	00050001
B+4	0	STO	0	00280000
B+5	0	DEL	0	002A0000
B+6	0	LOC	2	00020002
B+7	0	INT	2	00050002
B+8	0	STO	0	00280000
B+9	0	DEL	0	002A0000
B+10	0	LOC	3	00020003
B+11	0	INT	3	00050003
B+12	0	STO	0	00280000
B+13	0	DEL	0	002A0000
B+14	0	LOC	1	00020001
B+15	0	LOC	1	00020001
B+16	0	LOD	0	00270000
B+17	0	INT	4	00050004
B+18	0	MUL	0	00140000
B+19	0	STO	0	00280000
B+20	0	DEL	0	002A0000
B+21	0	LOC	1	00020001
B+22	0	LOC	1	00020001
B+23	0	LOD	0	00270000
B+24	0	LOC	2	00020002
B+25	0	LOD	0	00270000
B+26	0	ADD	0	00100000
B+27	0	STO	0	00280000
B+28	0	DEL	0	002A0000
B+29	0	LOC	2	00020002
B+30	0	LOC	2	00020002
B+31	0	LOD	0	00270000
B+32	0	LOC	3	00020003
B+33	0	LOD	0	00270000



B+34	0	ADD	0	00100000
B+35	0	STO	0	00280000
B+36	0	DEL	0	002A0000
B+37	0	LOC	2	00020002
B+38	0	LOC	2	00020002
B+39	0	LOD	0	00270000
B+40	0	INT	1	00050001
B+41	0	ADD	0	00100000
B+42	0	STO	0	00280000
B+43	0	DEL	0	002A0000
B+44	0	LOC	3	00020003
B+45	0	LOC	1	00020001
B+46	0	LOD	0	00270000
B+47	0	INT	3	00050003
B+48	0	MUL	0	00140000
B+49	0	STO	0	00280000
B+50	0	DEL	0	002A0000
B+51	0	LOC	3	00020003
B+52	0	LOC	3	00020003
B+53	0	LOD	0	00270000
B+54	0	INT	5	00050005
B+55	0	ADD	0	00100000
B+56	0	STO	0	00280000
B+57	0	DEL	0	002A0000
B+58	0	LOC	4	00020004
B+59	0	INT	6	00050006
B+60	0	STO	0	00280000
B+61	0	DEL	0	002A0000
B+62	0	LOC	1	00020001
B+63	0	LOD	0	00270000
B+64	0	LOC	2	00020002
B+65	0	LOD	0	00270000
B+66	0	GTR	0	00210000
B+67	0	XIT	79	0004004F
B+68	0	XIT	70	00040046
B+69	0	BSC	0	002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 2 IS:  
 C(88:91,\*,91) C(91:89,\*,96) C(89:90,\*,91) C(91:88,\*,92) C(96:90,\*,100)  
 C(100:92,\*,102) C(96:92,\*,104) A(4,93,93) A(3,102,102) A(2,92,92)  
 A(1,96,96)

CONTROL AT 70, FROM BLK 1 TO BLK 2 (PASS 1)				
C+70	0	ENT	C 92	0003805C
B+71	0	LOC	4	00020004
B+72	0	LOC	2	00020002
B+73	0	LOD	0	00270000
B+74	0	LOC	3	00020003
B+75	0	LOD	0	00270000
B+76	0	MUL	0	00140000
B+77	0	STO	0	00280000
B+78	0	DEL	0	002A0000
B+79	0	ENT	1	00030001

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 3 IS:  
 C(96:92,\*,104) C(100:92,\*,102) C(96:90,\*,100) C(91:88,\*,92) C(89:90,\*,91)  
 C(91:89,\*,96) C(88:91,\*,91) C(92:102,\*,106) A(4,106,106) A(3,102,102)  
 A(2,92,92) A(1,96,96)

CONTROL AT 79, FROM BLK 2 TO BLK 3 (PASS 1)				
C+79	0	ENT	C 97	00039061
B+80	0	LOC	1	00020001
B+81	0	LOC	3	00020003
B+82	0	LOD	0	00270000
B+83	0	LOC	2	00020002
B+84	0	LOD	0	00270000
B+85	0	MUL	0	00140000
B+86	0	STO	0	00280000
B+87	0	DEL	0	002A0000

# FINAL OPTIMIZATION RESULTS

#####

OPTIMIZATION AT 18	
(88,91 18,91)	CONSTANT PROPAGATION
OPTIMIZATION AT 26	
(89,91 26,96)	CONSTANT PROPAGATION
OPTIMIZATION AT 34	
(89,90 34,91)	CONSTANT PROPAGATION
OPTIMIZATION AT 41	
(88,91 41,92)	CONSTANT PROPAGATION



OPTIMIZATION AT 48  
(96,90|48,100)      CONSTANT PROPAGATION

OPTIMIZATION AT 55  
(92,100|55,102)      CONSTANT PROPAGATION

OPTIMIZATION AT 66  
(96,92|66,104)      CONSTANT PROPAGATION

OPTIMIZATION AT 76  
(102,92|76,106)      CONSTANT PROPAGATION

OPTIMIZATION AT 85  
(102,92|76,106)      COMM SUBEXP ELIM & CONS PROP

```

**** FORMATTED INTERMEDIATE CODE DUMP ****
LOC  OFF  OP CODE  CNS ADR  RAW CODE  OPTIMIZATION
SET                                     /ETC      TYPE
*****
0      0      ENT      88      00030058
1      0      TOGGLE   257     00080101
2      0      LOC      1      00020001
3      0      INT      1      00050001
4      0      STO      0      00280000
5      0      DEL      0      002A0000
6      0      LOC      2      00020002
7      0      INT      2      00050002
8      0      STO      0      00280000
9      0      DEL      0      002A0000
10     0      LOC      3      00020003
11     0      INT      3      00050003
12     0      STO      0      00280000
13     0      DEL      0      002A0000
14     0      LOC      1      00020001
15     0      LOC      1      00020001
16     0      LOD      0      00270000
17     0      INT      4      00050004
18     0      MUL      C 1024   00148400      CONSTANT PROPAGATION
19     0      STO      0      00280000
20     0      DEL      0      002A0000
21     0      LOC      1      00020001
22     0      LOC      1      00020001
23     0      LOD      0      00270000
24     0      LOC      2      00020002
25     0      LOD      0      00270000
26     0      ADD      C 1020   001083FC      CONSTANT PROPAGATION
27     0      STO      0      00280000
28     0      DEL      0      002A0000
29     0      LOC      2      00020002
30     0      LOC      2      00020002
31     0      LOD      0      00270000
32     0      LOC      3      00020003
33     0      LOD      0      00270000
34     0      ADD      C 1016   001083F8      CONSTANT PROPAGATION
35     0      STO      0      00280000
36     0      DEL      0      002A0000
37     0      LOC      2      00020002
38     0      LOC      2      00020002
39     0      LOD      0      00270000
40     0      INT      1      00050001
41     0      ADD      C 1012   001083F4      CONSTANT PROPAGATION
42     0      STO      0      00280000
43     0      DEL      0      002A0000
44     0      LOC      3      00020003
45     0      LOC      1      00020001
46     0      LOD      0      00270000
47     0      INT      3      00050003
48     0      MUL      C 1008   001483F0      CONSTANT PROPAGATION
49     0      STO      0      00280000
50     0      DEL      0      002A0000
51     0      LOC      3      00020003
52     0      LOC      3      00020003
53     0      LOD      0      00270000
54     0      INT      5      00050005
55     0      ADD      C 1004   001083EC      CONSTANT PROPAGATION
56     0      STO      0      00280000
57     0      DEL      0      002A0000
58     0      LOC      4      00020004
59     0      INT      6      00050006
60     0      STO      0      00280000
61     0      DEL      0      002A0000
62     0      LOC      1      00020001
63     0      LOD      0      00270000
64     0      LOC      2      00020002
65     0      LOD      0      00270000
66     0      GTR      C 1000   002183E8      CONSTANT PROPAGATION
67     0      XIT      79      0004004F

```





68	0	X11	70	00040048	
69	0	BSC	0	002F0000	
70	0	ENT	92	0003005C	
71	0	LOC	4	00020004	
72	0	LOC	2	00020002	
73	0	LOC	0	00270000	
74	0	LOC	3	00020003	
75	0	LOC	0	00270000	
76	0	MUL	C 982	001483D8	CONSTANT PROPAGATION
77	0	STO	0	00280000	
78	0	DEL	0	002A0000	
79	0	ENT	97	00030061	
80	0	LOC	1	00020001	
81	0	LOC	3	00020003	
82	0	LOC	0	00270000	
83	0	LOC	2	00020002	
84	0	LOC	0	00270000	
85	0	MUL	C 962	001483C2	COMM SUBEXP ELIM & CONS PROP
86	0	STO	0	00280000	
87	0	DEL	0	002A0000	

\*\*\*\*\*

FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

FINAL POOL FOR BLOCK# 1 IS:  
(NULL)

FINAL POOL FOR BLOCK# 2 IS:  
C(88:91,\*,91) C(91:89,\*,96) C(89:90,\*,91) C(91:88,\*,92) C(96:90,\*,100)  
C(100:92,\*,102) C(96:92,\*,104) A(4,93,93) A(3,102,102) A(2,92,92)  
A(1,96,96)

FINAL POOL FOR BLOCK# 3 IS:  
C(96:92,\*,104) C(100:92,\*,102) C(96:90,\*,100) C(91:88,\*,92) C(89:90,\*,91)  
C(91:89,\*,96) C(88:91,\*,91) C(92:102,\*,106) A(4,106,106) A(3,102,102)  
A(2,92,92) A(1,96,96)

BASIC BLOCK #1  
BEGINNING AT 0, ENDING AT 69  
BLOCK TRAVERSED 1 TIMES  
0 REFERENCES:  
BASIC BLOCK #2  
BEGINNING AT 70, ENDING AT 79  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
1  
BASIC BLOCK #3  
BEGINNING AT 79, ENDING AT 87  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
2

BLOCK SUMMARY DATA:

BLOCK NUMBER	NUMBER OF PASSES
1	1
2	1
3	1

TOTAL NUMBER OF BLOCKS PROCESSED WAS 3  
USING THE LAST-IN-FIRST-OUT BLOCK SELECTION ALGORITHM.

TIME FOR THE OPTIMIZATION WAS 0.295 SECONDS.

\*\*\*CODE SYNTHESIS FILTER IS COMPLETE\*\*\*





CARD	BL	SYL	COMMENT
1	0	0	TEST PROGRAM # 8 ;
3	0	1	\$EXECUTE BEGIN LOCAL A,B,C,D,E,F,G,H,I;
4	1	1	B := 5;
5	1	5	C := 10;
6	1	9	A := B * C;
7	1	17	IF A GTR B THEN
8	1	22	BEGIN
9	1	26	C := 30;
10	2	30	E := C;
11	2	35	G := B * E;
12	2	43	END ELSE
13	1	43	BEGIN
14	1	46	B := B + 2;
15	2	53	D := B;
16	2	58	F := D * C;
17	2	66	END;
18	1	67	H := C * B;
19	1	75	I := B * C;
20	1	83	WHILE H GTR C DO
21	1	93	BEGIN
22	1	93	C := C + 1;
23	2	100	F := 1 + 2;
24	2	106	G := G + 0;
25	2	113	A := A * 1;
26	2	120	B := B * 0;
27	2	127	END;
28	1	130	END
29	1	130	EOF

CODE FILE COPIED (130 WORDS)  
 CONSTANT TABLE COPIED (12 WORDS)  
 2 RECORDS WRITTEN INTO FILE 1  
 END OF COMPILATION FEBRUARY 12, 1975. CLOCK TIME = 20:50:41.39.

29 CARDS WERE READ.  
 NO ERRORS WERE DETECTED.



\*\*\*CODE SYNTHESIS FILTER\*\*\*

BLOCK SELECTION METHOD WILL BE STEEPEST DESCENT (MINIMUM CURRENT POOL)

\*\*\*COMPLETE TABLE DUMP\*\*\*  
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

\*\*\*TABLE DUMP\*\*\* AT LOCATION = 0

CONSTANTS	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
CONSYM	5	10	30	2	1	0								
CONTYPE	INT	INT	INT	INT	INT	INT								
CONVAL	131	132	133	134	135	136								
CONINT	5	10	30	2	1	0								

ADDR TABL	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
(EMPTY)														

VALUE STACK (TOP AT 127)

CONT STK	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
BLOCK#	0													
ENTRY	0													

EXEC STAC	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
(EMPTY)														

\*\*\* FORMATTED INTERMEDIATE CODE DUMP \*\*\*

LOC	OFF	OP CODE	CNS	ADR	RAW CODE	OPTIMIZATION
SET			/ETC			TYPE
0	0	ENT	C	131	00038083	
1	0	TGGGLE		257	00080101	
2	0	LOC		2	00020002	
3	0	INT		1	00050001	
4	0	STO		0	00280000	
5	0	DEL		0	002A0000	
6	0	LOC		3	00020003	
7	0	INT		2	00050002	
8	0	STO		0	00280000	
9	0	DEL		0	002A0000	
10	0	LOC		1	00020001	
11	0	LOC		2	00020002	
12	0	LOD		0	00270000	
13	0	LOC		3	00020003	
14	0	LOD		0	00270000	
15	0	MUL		0	00140000	
16	0	STO		0	00280000	
17	0	DEL		0	002A0000	
18	0	LOC		1	00020001	
19	0	LOD		0	00270000	
20	0	LOC		2	00020002	
21	0	LOD		0	00270000	
22	0	GTR		0	00210000	
23	0	XIT	46		0004002E	
24	0	XIT	26		0004001A	
25	0	BSC	06		002F0000	
26	0	ENT	1		00030001	
27	0	LOC	3		00020003	
28	0	INT	3		00050003	
29	0	STO	0		00280000	
30	0	DEL	0		002A0000	
31	0	LOC	5		00020005	
32	0	LOC	3		00020003	
33	0	LOD	0		00270000	
34	0	STO	0		00280000	
35	0	DEL	0		002A0000	
36	0	LOC	7		00020007	
37	0	LOC	2		00020002	
38	0	LOD	0		00270000	
39	0	LOC	5		00020005	
40	0	LOD	0		00270000	
41	0	MUL	0		00140000	



42	0	STU	0	00280000
43	0	DEL	0	002A0000
44	0	XIT	67	00040043
45	0	BRS	0	002E0000
46	0	ENT	1	00030001
47	0	LOC	2	00020002
48	0	LOC	2	00020002
49	0	LOC	0	00270000
50	0	INT	4	00050004
51	0	ADD	0	00100000
52	0	STO	0	00280000
53	0	DEL	0	002A0000
54	0	LOC	4	00020004
55	0	LOC	2	00020002
56	0	LOC	0	00270000
57	0	STO	0	00280000
58	0	DEL	0	002A0000
59	0	LOC	6	00020006
60	0	LOC	4	00020004
61	0	LOC	0	00270000
62	0	LOC	3	00020003
63	0	LOC	0	00270000
64	0	MUL	0	00140000
65	0	STO	0	00280000
66	0	DEL	0	002A0000
67	0	ENT	1	00030001
68	0	LOC	8	00020008
69	0	LOC	3	00020003
70	0	LOC	0	00270000
71	0	LOC	2	00020002
72	0	LOC	0	00270000
73	0	MUL	0	00140000
74	0	STO	0	00280000
75	0	DEL	0	002A0000
76	0	LOC	9	00020009
77	0	LOC	2	00020002
78	0	LOC	0	00270000
79	0	LOC	3	00020003
80	0	LOC	0	00270000
81	0	MUL	0	00140000
82	0	STO	0	00280000
83	0	DEL	0	002A0000
84	0	ENT	1	00030001
85	0	LOC	8	00020008
86	0	LOC	0	00270000
87	0	LOC	3	00020003
88	0	LOC	0	00270000
89	0	GTR	0	00210000
90	0	XIT	130	00040082
91	0	XIT	93	00040050
92	0	BSC	0	002F0000
93	0	ENT	1	00030001
94	0	LOC	3	00020003
95	0	LOC	3	00020003
96	0	LOC	0	00270000
97	0	INT	5	00050005
98	0	ADD	0	00100000
99	0	STO	0	00280000
100	0	DEL	0	002A0000
101	0	LOC	6	00020006
102	0	INT	5	00050005
103	0	INT	4	00050004
104	0	ADD	0	00100000
105	0	STO	0	00280000
106	0	DEL	0	002A0000
107	0	LOC	7	00020007
108	0	LOC	7	00020007
109	0	LOC	0	00270000
110	0	INT	6	00050006
111	0	ADD	0	00100000
112	0	STO	0	00280000
113	0	DEL	0	002A0000
114	0	LOC	1	00020001
115	0	LOC	1	00020001
116	0	LOC	0	00270000
117	0	INT	5	00050005
118	0	MUL	0	00140000
119	0	STO	0	00280000
120	0	DEL	0	002A0000
121	0	LOC	2	00020002
122	0	LOC	2	00020002
123	0	LOC	0	00270000
124	0	INT	6	00050006
125	0	MUL	0	00140000
126	0	STO	0	00280000
127	0	DEL	0	002A0000
128	0	XIT	84	00040054
129	0	BRS	0	002E0000



130 0 1ENI 1 11 1 00030001 1  
 \*\*\*\*\*

BASIC BLOCK #1  
 BEGINNING AT 0, ENDING AT 0  
 BLOCK TRAVERSED 0 TIMES  
 0 REFERENCES:

\*\*\*END OF COMPLETE TABLE DUMP\*\*\*

B+2	0	LOC	2	00020002
B+3	0	INT	1	00050001
B+4	0	STO	0	00280000
B+5	0	DEL	0	002A0000
B+6	0	LOC	3	00020003
B+7	0	INT	2	00050002
B+8	0	STO	0	00280000
B+9	0	DEL	0	002A0000
B+10	0	LOC	1	00020001
B+11	0	LOC	2	00020002
B+12	0	LOC	0	00270000
B+13	0	LOC	3	00020003
B+14	0	LOC	0	00270000
B+15	0	MUL	0	00140000
B+16	0	STO	0	00280000
B+17	0	DEL	0	002A0000
B+18	0	LOC	1	00020001
B+19	0	LOC	0	00270000
B+20	0	LOC	2	00020002
B+21	0	LOC	0	00270000
B+22	0	GTR	0	00210000
B+23	0	XIT	46	0004002E
B+24	0	XIT	26	0004001A
B+25	0	BSC	0	002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 2 IS:  
 C(131:132,\*,138) C(138:131,\*,140) A(1,138,138) A(3,132,132) A(2,131,131)

CONTROL AT 26, FROM BLK 1 TO BLK 2 (PASS 1)

C+26	0	ENT	C 135	00038087
B+27	0	LOC	3	00020003
B+28	0	INT	3	00050003
B+29	0	STO	0	00280000
B+30	0	DEL	0	002A0000
B+31	0	LOC	5	00020005
B+32	0	LOC	3	00020003
B+33	0	LOC	0	00270000
B+34	0	STO	0	00280000
B+35	0	DEL	0	002A0000
B+36	0	LOC	7	00020007
B+37	0	LOC	2	00020002
B+38	0	LOC	0	00270000
B+39	0	LOC	5	00020005
B+40	0	LOC	0	00270000
B+41	0	MUL	0	00140000
B+42	0	STO	0	00280000
B+43	0	DEL	0	002A0000
B+44	0	XIT	67	00040043
B+45	0	BRS	0	002E0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 3 IS:  
 C(138:131,\*,140) C(131:132,\*,138) C(131:133,\*,142) A(7,142,142)  
 A(5,133,133) A(3,133,133) A(1,133,138) A(2,131,131)

CONTROL AT 67, FROM BLK 2 TO BLK 3 (PASS 1)

C+67	0	ENT	C 140	0003808C
B+68	0	LOC	8	00020008
B+69	0	LOC	3	00020003
B+70	0	LOC	0	00270000
B+71	0	LOC	2	00020002
B+72	0	LOC	0	00270000
B+73	0	MUL	0	00140000
B+74	0	STO	0	00280000
B+75	0	DEL	0	002A0000
B+76	0	LOC	9	00020009
B+77	0	LOC	2	00020002
B+78	0	LOC	0	00270000
B+79	0	LOC	3	00020003
B+80	0	LOC	0	00270000
B+81	0	MUL	0	00140000
B+82	0	STO	0	00280000
B+83	0	DEL	0	002A0000
B+84	0	ENT	1	00030001

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 4 IS:  
 C(131:133,\*,142) C(131:132,\*,138) C(138:131,\*,140) A(9,142,142)  
 A(8,142,142) A(7,142,142) A(5,133,133) A(3,133,133) A(1,138,138)





A(2,131,131)

CONTROL	AT	84,	FROM BLK 3	TO BLK 4	(PASS 1)
C+84	0	ENT	C	145	00038091
B+85	0	LOC		8	00020003
B+86	0	LOC		0	00270000
B+87	0	LOC		3	00020003
B+88	0	LOC		0	00270000
B+89	0	GTR		0	00210000
B+90	0	XIT		130	00040082
B+91	0	XIT		93	00040050
B+92	0	BSC		0	002F0000

INITIAL CURRENT OPTIMIZED PCOL TO BE USED FOR BLOCK# 5 IS:

C(138:131,\*,140) C(131:132,\*,138) C(131:133,\*,142) C(142:133,\*,140)  
 A(9,142,142) A(8,142,142) A(7,142,142) A(5,133,133) A(3,133,133)  
 A(1,138,138) A(2,131,131)

CONTROL	AT	93,	FROM BLK 4	TO BLK 5	(PASS 1)
C+93	0	ENT	C	150	00038096
B+94	0	LOC		3	00020003
B+95	0	LOC		3	00020003
B+96	0	LOC		0	00270000
B+97	0	INT		5	00050005
B+98	0	ADD		0	00100000
B+99	0	STO		0	00280000
B+100	0	DEL		0	002A0000
B+101	0	LOC		6	00020006
B+102	0	INT		5	00050005
B+103	0	INT		4	00050004
B+104	0	ADD		0	00100000
B+105	0	STO		0	00280000
B+106	0	DEL		0	002A0000
B+107	0	LOC		7	00020007
B+108	0	LOC		7	00020007
B+109	0	LOC		0	00270000
B+110	0	INT		6	00050006
B+111	0	ADD		0	00100000
B+112	0	STO		0	00280000
B+113	0	DEL		0	002A0000
B+114	0	LOC		1	00020001
B+115	0	LOC		1	00020001
B+116	0	LOC		0	00270000
B+117	0	INT		5	00050005
B+118	0	MUL		0	00140000
B+119	0	STO		0	00280000
B+120	0	DEL		0	002A0000
B+121	0	LOC		2	00020002
B+122	0	LOC		2	00020002
B+123	0	LOC		0	00270000
B+124	0	INT		6	00050006
B+125	0	MUL		0	00140000
B+126	0	STO		0	00280000
B+127	0	DEL		0	002A0000
B+128	0	XIT		84	00040054
B+129	0	BRS		0	002E0000

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 5 IS:

C(142:133,\*,140) C(131:133,\*,142) C(131:132,\*,138) C(138:131,\*,140)  
 C(133:135,\*,145) C(131:133,\*,145) C(135:134,\*,147) C(142:136,\*,142)  
 C(138:135,\*,138) C(131:136,\*,136) A(2,136,136) A(1,138,138) A(7,142,142)  
 A(6,147,147) A(3,145,145) A(9,142,142) A(8,142,142) A(5,133,133)

CURRENT POOL FOR BLOCK# 4 IS:

C(131:133,\*,142) C(131:132,\*,138) C(138:131,\*,140) A(9,142,142)  
 A(8,142,142) A(7,142,142) A(5,133,133) A(3,133,133) A(1,138,138)  
 A(2,131,131)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 4 IS:

A(9,142,142) A(8,142,142) A(7,142,142) A(5,133,133) A(3,\*,151)  
 A(1,138,138) A(2,\*,152)

CONTROL	AT	84,	FROM BLK 5	TO BLK 4	(PASS 2)
C+84	0	ENT	C	145	00038091
B+85	0	LOC		8	00020008
B+86	0	LOC		0	00270000
B+87	0	LOC		3	00020003
B+88	0	LOC		0	00270000
B+89	0	GTR	C	988	002183DC
B+90	0	XIT		130	00040082
B+91	0	XIT		93	00040050
B+92	0	BSC		0	002F0000

CONSTANT PROPAGATION



OPTIMIZATION AT 15  
(131,132|15,138) CONSTANT PROPAGATION

OPTIMIZATION AT 22  
(138,131|22,140) CONSTANT PROPAGATION

OPTIMIZATION AT 41  
(131,133|41,142) CONSTANT PROPAGATION

OPTIMIZATION AT 73  
(131,133|41,142) COMM SUBEXP ELIM & CONS PROP

OPTIMIZATION AT 81  
(131,133|41,142) COMM SUBEXP ELIM & CONS PROP

OPTIMIZATION AT 104  
(134,135|104,147) CONSTANT PROPAGATION

OPTIMIZATION AT 111  
(136,142|111,142) CONSTANT PROPAGATION

OPTIMIZATION AT 118  
(135,138|118,138) CONSTANT PROPAGATION

OPTIMIZATION AT 125  
(136,155|125,136) SIMPLIFICATION

```

**** FORMATTED INTERMEDIATE CODE DUMP ****
LOC  OFF  OP CODE  CNS ADR  RAW CODE  OPTIMIZATION
  SET  /ETC      TYPE
*****
0      0      ENT      131      00030083
1      0      TOGGLE   257      00060101
2      0      LOC      2       00020002
3      0      INT      1       00050001
4      0      STO      0       00280000
5      0      DEL      0       002A0000
6      0      LOC      3       00020003
7      0      INT      2       00050002
8      0      STO      0       00280000
9      0      DEL      0       002A0000
10     0      LOC      1       00020001
11     0      LOC      2       00020002
12     0      LOD      0       00270000
13     0      LOC      3       00020003
14     0      LOD      0       00270000
15     0      MUL      C 1024    00148400      CONSTANT PROPAGATION
16     0      STO      0       00280000
17     0      DEL      0       002A0000
18     0      LOC      1       00020001
19     0      LOD      0       00270000
20     0      LOC      2       00020002
21     0      LOD      0       00270000
22     0      GTR      C 1020    002183FC      CONSTANT PROPAGATION
23     0      XIT      46      0004002E
24     0      XIT      26      0004001A
25     0      BSC      0       002F0000
26     0      ENT      135      00030087
27     0      LOC      3       00020003
28     0      INT      3       00050003
29     0      STO      0       00280000
30     0      DEL      0       002A0000
31     0      LOC      5       00020005
32     0      LOC      3       00020003
33     0      LOD      0       00270000
34     0      STO      0       00280000
35     0      DEL      0       002A0000
36     0      LOC      7       00020007
37     0      LOC      2       00020002
38     0      LOD      0       00270000
39     0      LOC      5       00020005
40     0      LOD      0       00270000
41     0      MUL      C 1012    001483F4      CONSTANT PROPAGATION
42     0      STO      0       00280000
43     0      DEL      0       002A0000
44     0      XIT      67      00040043
45     0      BRS      0       002E0000
46     0      ENT      1       00030001
47     0      LOC      2       00020002
48     0      LOC      2       00020002
49     0      LOD      0       00270000
50     0      INT      4       00050004
51     0      ADD      0       00100000
52     0      STO      0       00280000
53     0      DEL      0       002A0000
54     0      LOC      4       00020004
55     0      LOC      2       00020002

```



56	0	LDU	0	00270000	
57	0	STO	0	00280000	
58	0	DEL	0	002A0000	
59	0	LOC	6	00020006	
60	0	LOC	4	00020004	
61	0	LOD	0	00270000	
62	0	LOC	3	00020003	
63	0	LOD	0	00270000	
64	0	MUL	0	00140000	
65	0	STO	0	00280000	
66	0	DEL	0	002A0000	
67	0	ENT	140	0003008C	
68	0	LOC	8	00020008	
69	0	LOC	3	00020003	
70	0	LOD	0	00270000	
71	0	LOC	2	00020002	
72	0	LOD	0	00270000	
73	0	MUL	C 1002	001483EA	COMM SUBEXP ELIM & CONS PROP
74	0	STO	0	00280000	
75	0	DEL	0	002A0000	
76	0	LOC	9	00020009	
77	0	LOC	2	00020002	
78	0	LOD	0	00270000	
79	0	LOC	3	00020003	
80	0	LOD	0	00270000	
81	0	MUL	C 998	001483E6	COMM SUBEXP ELIM & CONS PROP
82	0	STO	0	00280000	
83	0	DEL	0	002A0000	
84	0	ENT	145	00030091	
85	0	LOC	8	00020008	
86	0	LOD	0	00270000	
87	0	LOC	3	00020003	
88	0	LOD	0	00270000	
89	0	GTR	988	002103DC	
90	0	XIT	130	00040082	
91	0	XIT	93	0004005D	
92	0	BSC	0	002F0000	
93	0	ENT	150	00030096	
94	0	LOC	3	00020003	
95	0	LOC	3	00020003	
96	0	LOD	0	00270000	
97	0	INT	5	00050005	
98	0	ADD	976	001003D0	
99	0	STO	0	00280000	
100	0	DEL	0	002A0000	
101	0	LOC	6	00020006	
102	0	INT	5	00050005	
103	0	INT	4	00050004	
104	0	ADD	C 972	001083CC	CONSTANT PROPAGATION
105	0	STO	0	00280000	
106	0	DEL	0	002A0000	
107	0	LOC	7	00020007	
108	0	LOC	7	00020007	
109	0	LOD	0	00270000	
110	0	INT	6	00050006	
111	0	ADD	C 968	001083C8	CONSTANT PROPAGATION
112	0	STO	0	00280000	
113	0	DEL	0	002A0000	
114	0	LOC	1	00020001	
115	0	LOC	1	00020001	
116	0	LOD	0	00270000	
117	0	INT	5	00050005	
118	0	MUL	C 964	001483C4	CONSTANT PROPAGATION
119	0	STO	0	00280000	
120	0	DEL	0	002A0000	
121	0	LOC	2	00020002	
122	0	LOC	2	00020002	
123	0	LOD	0	00270000	
124	0	INT	6	00050006	
125	0	MUL	C 960	001483C0	SIMPLIFICATION
126	0	STO	0	00280000	
127	0	DEL	0	002A0000	
128	0	XIT	84	00040054	
129	0	BRS	0	002E0000	
130	0	ENT	155	0003009B	

\*\*\*\*\*

FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

FINAL POOL FOR BLOCK# 1 IS:  
(NULL)





FINAL POCL FOR BLOCK# 2 IS:  
C(131:132,\*,138) C(138:131,\*,140) A(1,138,138) A(3,132,132) A(2,131,131)

FINAL POCL FOR BLOCK# 3 IS:  
C(138:131,\*,140) C(131:132,\*,138) C(131:133,\*,142) A(7,142,142)  
A(5,133,133) A(3,133,133) A(1,138,138) A(2,131,131)

FINAL POCL FOR BLOCK# 4 IS:  
A(9,142,142) A(8,142,142) A(7,142,142) A(5,133,133) A(3,\*,160)  
A(1,138,138) A(2,\*,161)

FINAL POCL FOR BLOCK# 5 IS:  
A(9,142,142) A(8,142,142) A(7,142,142) A(5,133,133) A(3,\*,154)  
A(1,138,138) A(2,\*,155) C(142:154,\*,156)

FINAL POCL FOR BLOCK# 6 IS:  
C(142:151,\*,153) A(9,142,142) A(8,142,142) A(7,142,142) A(5,133,133)  
A(3,\*,151) A(1,138,138) A(2,\*,152)

BASIC BLOCK #1  
BEGINNING AT 0, ENDING AT 25  
BLOCK TRAVERSED 1 TIMES  
0 REFERENCES:  
BASIC BLOCK #2  
BEGINNING AT 26, ENDING AT 45  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
1  
BASIC BLOCK #3  
BEGINNING AT 67, ENDING AT 84  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
2  
BASIC BLOCK #4  
BEGINNING AT 84, ENDING AT 92  
BLOCK TRAVERSED 2 TIMES  
1 REFERENCES:  
3  
BASIC BLOCK #5  
BEGINNING AT 93, ENDING AT 129  
BLOCK TRAVERSED 2 TIMES  
1 REFERENCES:  
4  
BASIC BLOCK #6  
BEGINNING AT 130, ENDING AT 130  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
4

# BLOCK SUMMARY DATA:

BLOCK NUMBER	NUMBER OF PASSES
1	1
2	1
3	1
4	2
5	2
6	1

TOTAL NUMBER OF BLOCKS PROCESSED WAS 8  
USING THE STEEPEST DESCENT (MINIMUM CURRENT POOL) BLOCK SELECTION ALGORITHM.  
TIME FOR THE OPTIMIZATION WAS 0.460 SECONDS.

\*\*\*CODE SYNTHESIS FILTER IS COMPLETE\*\*\*





CARD	BL	SYL	COMMENT
1	0	0	TEST PROGRAM # 9 ;
2	0	1	BEGIN LOCAL A,B,C,D,E,F,G,H,I,J,K;
3	0	1	READ(A, B, C);
4	1	10	I := B * C;
5	1	18	J := C * B;
6	1	26	WHILE A GTR B DO
7	1	36	BEGIN
8	1	36	WHILE A GTR B DO
9	1	46	BEGIN
10	2	46	WHILE A GTR B DO
11	2	56	B := B + 1;
12	3	66	J := B * C;
13	3	74	K := 1;
14	3	78	G := 1 + 3;
15	3	84	END;
16	3	87	END;
17	2	90	END
18	1	90	EOF
19	1	90	EOF

CODE FILE COPIED (90 WORDS)  
 CONSTANT TABLE COPIED (4 WORDS)  
 2 RECORDS WRITTEN INTO FILE 1  
 END OF COMPILATION FEBRUARY 12, 1975. CLOCK TIME = 20:34:0.02.

19 CARDS WERE READ.  
 NO ERRORS WERE DETECTED.



\*\*\*CODE SYNTHESIS FILTER\*\*\*

BLOCK SELECTION METHOD WILL BE FIRST-IN-FIRST-OUT

\*\*\*COMPLETE TABLE DUMP\*\*\*  
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

\*\*\*TABLE DUMP\*\*\* AT LOCATION = 0

CONSTANTS	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
CONSYM	1	2	3	4										
CONTYPE	INT	INT	INT	INT										
CONVAL	100	101	102	103										
CONINT	1	2	3	4										

ADDR TABL	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
(EMPTY)														

VALUE STACK (TOP AT 127)

CONT STK	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
BLOCK#	0													
ENTRY	0													

EXEC STAC	<1>	<2>	<3>	<4>	<5>	<6>	<7>	<8>	<9>	<10>	<11>	<12>	<13>	<14>
(EMPTY)														

\*\*\*\* FORMATTED INTERMEDIATE CODE DUMP \*\*\*\*

LOC	OFF	OP CODE	CNS ADR	RAW CODE	OPTIMIZATION
SET			/ETC		TYPE
0	0	ENT	C 100	00038064	
1	0	TGGLE	257	00080101	
2	0	LOC	1	00020001	
3	0	INT	0	00050000	
4	0	STD	0	00290000	
5	0	LOC	2	00020002	
6	0	INT	0	00050000	
7	0	STD	0	00290000	
8	0	LOC	3	00020003	
9	0	INT	1	00050001	
10	0	STD	0	00280000	
11	0	DEL	0	002A0000	
12	0	LOC	1	00020001	
13	0	LOD	0	00270000	
14	0	LOC	2	00020002	
15	0	LOD	0	00270000	
16	0	GTR	0	00210000	
17	0	XIT	73	00040049	
18	0	XIT	20	00040014	
19	0	BSC	0	002F0000	
20	0	ENT	1	00030001	
21	0	LOC	1	00020001	
22	0	LOD	0	00270000	
23	0	LOC	2	00020002	
24	0	LOD	0	00270000	
25	0	GTR	0	00210000	
26	0	XIT	44	0004002C	
27	0	XIT	29	0004001D	
28	0	BSC	0	002F0000	
29	0	ENT	1	00030001	
30	0	LOC	2	00020002	
31	0	INT	2	00050002	
32	0	STD	0	00280000	
33	0	DEL	0	002A0000	
34	0	LOC	4	00020004	
35	0	LOC	2	00020002	
36	0	LOD	0	00270000	
37	0	LOC	3	00020003	
38	0	LOD	0	00270000	
39	0	MUL	0	00140000	
40	0	STD	0	00280000	
41	0	DEL	0	002A0000	



42	0	XIT	5	00040039
43	0	BRS	0	002E0000
44	0	ENT	1	00030001
45	0	LOC	2	00020002
46	0	INT	5	00050003
47	0	STO	0	00280000
48	0	DEL	0	002A0000
49	0	LOC	8	00020008
50	0	LOC	2	00020002
51	0	LOD	0	00270000
52	0	LOC	3	00020003
53	0	LOD	0	00270000
54	0	MUL	0	00140000
55	0	STO	0	00280000
56	0	DEL	0	002A0000
57	0	ENT	1	00030001
58	0	LOC	5	00020005
59	0	LOC	2	00020002
60	0	LOD	0	00270000
61	0	STO	0	00280000
62	0	DEL	0	002A0000
63	0	LOC	6	00020006
64	0	LOC	5	00020005
65	0	LOD	0	00270000
66	0	LOC	3	00020003
67	0	LOD	0	00270000
68	0	MUL	0	00140000
69	0	STO	0	00280000
70	0	DEL	0	002A0000
71	0	XIT	86	00040056
72	0	BRS	0	002E0000
73	0	ENT	1	00030001
74	0	LOC	2	00020002
75	0	INT	4	00050004
76	0	STO	0	00280000
77	0	DEL	0	002A0000
78	0	LOC	8	00020008
79	0	LOC	2	00020002
80	0	LOD	0	00270000
81	0	LOC	3	00020003
82	0	LOD	0	00270000
83	0	MUL	0	00140000
84	0	STO	0	00280000
85	0	DEL	0	002A0000
86	0	ENT	1	00030001
87	0	LOC	7	00020007
88	0	LOC	2	00020002
89	0	LOD	0	00270000
90	0	STO	0	00280000
91	0	DEL	0	002A0000
92	0	LOC	7	00020007
93	0	LOC	7	00020007
94	0	LOD	0	00270000
95	0	LOC	3	00020003
96	0	LOD	0	00270000
97	0	MUL	0	00140000
98	0	STO	0	00280000
99	0	DEL	0	002A0000

\*\*\*\*\*

BASIC BLOCK #1  
 BEGINNING AT 0, ENDING AT 0  
 BLOCK TRAVERSED 0 TIMES  
 0 REFERENCES:

\*\*\*END OF COMPLETE TABLE DUMP\*\*\*

B+2	0	LOC	1	00020001
B+3	0	INT	0	00050000
B+4	0	STO	0	00290000
B+5	0	LOC	2	00020002
B+6	0	INT	0	00050000
B+7	0	STO	0	00290000
B+8	0	LOC	3	00020003
B+9	0	INT	1	00050001
B+10	0	STO	0	00280000
B+11	0	DEL	0	002A0000
B+12	0	LOC	1	00020001
B+13	0	LOD	0	00270000
B+14	0	LOC	2	00020002
B+15	0	LOD	0	00270000
B+16	0	GTR	0	00210000
B+17	0	XIT	73	00040049
B+18	0	XIT	20	00040014
B+19	0	BSC	0	002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 2 IS:



C(104:105,\*,106) A(3,100,100) A(2,\*,105) A(1,\*,104)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 3 IS:  
C(104:105,\*,106) A(3,100,100) A(2,\*,105) A(1,\*,104)

CONTROL AT 73, FROM BLK 1 TO BLK 2 (PASS 1)			
C+73	0	ENT	C 104 00038068
B+74	0	LOC	2 00020002
B+75	0	INT	4 00050004
B+76	0	STO	0 00280000
B+77	0	DEL	0 002A0000
B+78	0	LOC	8 00020008
B+79	0	LOC	2 00020002
B+80	0	LOD	0 00270000
B+81	0	LOC	3 00020003
B+82	0	LOD	0 00270000
B+83	0	MUL	0 00140000
B+84	0	STO	0 00280000
B+85	0	DEL	0 002A0000
B+86	0	ENT	1 00030001

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 4 IS:  
C(104:105,\*,106) C(105:100,\*,103) A(8,103,103) A(2,103,103) A(3,100,100)  
A(1,\*,104)

CONTROL AT 20, FROM BLK 1 TO BLK 3 (PASS 1)			
C+20	0	ENT	C 109 00038060
B+21	0	LOC	1 00020001
B+22	0	LOD	0 00270000
B+23	0	LOC	2 00020002
B+24	0	LOD	0 00270000
B+25	0	GTR	0 00210000
B+26	0	XIT	44 0004002C
B+27	0	XIT	29 0004001D
B+28	0	BSC	0 002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 5 IS:  
C(104:105,\*,106) A(3,100,100) A(2,\*,105) A(1,\*,104)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 6 IS:  
C(104:105,\*,106) A(3,100,100) A(2,\*,105) A(1,\*,104)

CONTROL AT 86, FROM BLK 2 TO BLK 4 (PASS 1)			
C+86	0	ENT	C 114 00038072
B+87	0	LOC	7 00020007
B+88	0	LOC	2 00020002
B+89	0	LOD	0 00270000
B+90	0	STO	0 00280000
B+91	0	DEL	0 002A0000
B+92	0	LOC	7 00020007
B+93	0	LOC	7 00020007
B+94	0	LOD	0 00270000
B+95	0	LOC	3 00020003
B+96	0	LOD	0 00270000
B+97	0	MUL	0 00140000
B+98	0	STO	0 00280000
B+99	0	DEL	0 002A0000

CONTROL AT 44, FROM BLK 3 TO BLK 5 (PASS 1)			
C+44	0	ENT	C 119 00038077
B+45	0	LOC	2 00020002
B+46	0	INT	3 00050003
B+47	0	STO	0 00280000
B+48	0	DEL	0 002A0000
B+49	0	LOC	8 00020008
B+50	0	LOC	2 00020002
B+51	0	LOD	0 00270000
B+52	0	LOC	3 00020003
B+53	0	LOD	0 00270000
B+54	0	MUL	0 00140000
B+55	0	STO	0 00280000
B+56	0	DEL	0 002A0000
B+57	0	ENT	1 00030001

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 7 IS:  
C(104:105,\*,106) C(102:100,\*,102) A(8,102,102) A(2,102,102) A(3,100,100)  
A(1,\*,104)

CONTROL AT 29, FROM BLK 3 TO BLK 6 (PASS 1)			
C+29	0	ENT	C 124 0003807C
B+30	0	LOC	2 00020002
B+31	0	INT	2 00050002
B+32	0	STO	0 00280000
B+33	0	DEL	0 002A0000
B+34	0	LOC	4 00020004
B+35	0	LOC	2 00020002
B+36	0	LOD	0 00270000





B+37	0	LOC	3	00020003
B+38	0	LOC	0	00270000
B+39	0	MUL	0	00140000
B+40	0	STO	0	00280000
B+41	0	DEL	0	002A0000
B+42	0	XIT	57	00040039
B+43	0	BRS	0	002E0000

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 6 IS:

C(104:105,\*,106) C(101:100,\*,101) A(4,101,101) A(2,101,101) A(3,100,100)  
A(1,\*,104)

CURRENT POOL FOR BLOCK# 7 IS:

C(104:105,\*,106) C(102:100,\*,102) A(8,102,102) A(2,102,102) A(3,100,100)  
A(1,\*,104)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 7 IS:

A(2,\*,110) A(3,100,100) A(1,\*,104) C(110:100,\*,110)

CONTROL AT 57, FROM BLK 6 TO BLK 7 (PASS 1)				
C+57	0	ENT	C 129	00038081
B+58	0	LOC	5	00020005
B+59	0	LOC	2	00020002
B+60	0	LOC	0	00270000
B+61	0	STO	0	00280000
B+62	0	DEL	0	002A0000
B+63	0	LOC	6	00020006
B+64	0	LOC	5	00020005
B+65	0	LOC	0	00270000
B+66	0	LOC	3	00020003
B+67	0	LOC	0	00270000
B+68	0	MUL	0	00140000
B+69	0	STO	0	00280000
B+70	0	DEL	0	002A0000
B+71	0	XIT	86	00040056
B+72	0	BRS	0	002E0000

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 7 IS:

C(110:100,\*,110) A(6,\*,110) A(5,\*,110) A(2,\*,110) A(3,100,100)  
A(1,\*,104)

CURRENT POOL FOR BLOCK# 4 IS:

C(104:105,\*,106) C(103:100,\*,103) A(8,103,103) A(2,103,103) A(3,100,100)  
A(1,\*,104)

AFTER PERFORMING THE MEET OPERATION:

NEW CURRENT POOL/INPUT POOL FOR BLOCK# 4 IS:

A(2,\*,111) A(3,100,100) A(1,\*,104) C(111:100,\*,111)

CONTROL AT 86, FROM BLK 7 TO BLK 4 (PASS 2)				
C+86	0	ENT	C 114	00038072
B+87	0	LOC	7	00020007
B+88	0	LOC	2	00020002
B+89	0	LOC	0	00270000
B+90	0	STO	0	00280000
B+91	0	DEL	0	002A0000
B+92	0	LOC	7	00020007
B+93	0	LOC	7	00J20007
B+94	0	LOC	0	00270000
B+95	0	LOC	3	00020003
B+96	0	LOC	0	00270000
B+97	0	MUL	C 1000	001483E8
B+98	0	STO	0	00280000
B+99	0	DEL	0	002A0000

COMM SUBEXP ELIM & CONS PROP

# FINAL OPTIMIZATION RESULTS

#####

OPTIMIZATION AT 25

(104,105|16,106)

COMM SUBEXP ELIM

OPTIMIZATION AT 39

(100,101|39,101)

CONSTANT PROPAGATION

OPTIMIZATION AT 51



OPTIMIZATION AT 34  
(102,100|54,102)      CONSTANT PROPAGATION

OPTIMIZATION AT 68  
(110,100|68,110)      SIMPLIFICATION

OPTIMIZATION AT 83  
(100,103|83,103)      CONSTANT PROPAGATION

OPTIMIZATION AT 97  
(100,111|97,111)      SIMPLIFICATION

```

**** FORMATTED INTERMEDIATE CODE DUMP ****
LOC OFF  OP CODE  CNS ADR  RAW CODE  OPTIMIZATION
SET                               /ETC      TYPE
*****
0      0      ENT      100      00030084
1      0      TOGGLE   257      00080101
2      0      LOC      1       00020001
3      0      INT      0       00050000
4      0      STD      0       00290000
5      0      LOC      2       00020002
6      0      INT      0       00050000
7      0      STD      0       00290000
8      0      LOC      3       00020003
9      0      INT      1       00050001
10     0      STO      0       00280000
11     0      DEL      0       002A0000
12     0      LOC      1       00020001
13     0      LOD      0       00270000
14     0      LOC      2       00020002
15     0      LOD      0       00270000
16     0      GTR      1024     00210400
17     0      XIT      73      00040049
18     0      XIT      20      00040014
19     0      BSC      0       002F0000
20     0      ENT      109     0003006D
21     0      LOC      1       00020001
22     0      LOD      0       00270000
23     0      LOC      2       00020002
24     0      LOD      0       00270000
25     0      GTR      C 1008   002183F0      COMM SUBEXP ELIM
26     0      XIT      44      0004002C
27     0      XIT      29      0004001D
28     0      BSC      0       002F0000
29     0      ENT      124     0003007C
30     0      LOC      2       00020002
31     0      INT      2       00050002
32     0      STO      0       00280000
33     0      DEL      0       002A0000
34     0      LOC      4       00020004
35     0      LOC      2       00020002
36     0      LOD      0       00270000
37     0      LOC      3       00020003
38     0      LOD      0       00270000
39     0      MUL      C 988    001483DC      CONSTANT PROPAGATION
40     0      STO      0       00280000
41     0      DEL      0       002A0000
42     0      XIT      57      00040039
43     0      BRS      0       002E0000
44     0      ENT      119     00030077
45     0      LOC      2       00020002
46     0      INT      3       00050003
47     0      STO      0       00280000
48     0      DEL      0       002A0000
49     0      LOC      8       00020008
50     0      LOC      2       00020002
51     0      LOD      0       00270000
52     0      LOC      3       00020003
53     0      LOD      0       00270000
54     0      MUL      C 996    001483E4      CONSTANT PROPAGATION
55     0      STO      0       00280000
56     0      DEL      0       002A0000
57     0      ENT      129     00030081
58     0      LOC      5       00020005
59     0      LOC      2       00020002
60     0      LOD      0       00270000
61     0      STO      0       00280000
62     0      DEL      0       002A0000
63     0      LOC      6       00020006
64     0      LOC      5       00020005
65     0      LOD      0       00270000
66     0      LOC      3       00020003
67     0      LOD      0       00270000
68     0      MUL      C 984    001483D8      SIMPLIFICATION
69     0      STO      0       00280000
70     0      DEL      0       002A0000

```



71	U	X11		86	00040056	
72	0	BRS		0	002E0000	
73	0	ENT		104	00030068	
74	0	LOC		2	00020002	
75	0	INT		4	00050004	
76	0	STO		0	00280000	
77	0	DEL		0	002A0000	
78	0	LOC		8	00020008	
79	0	LOC		2	00020002	
80	0	LOD		0	00270000	
81	0	LOC		3	00020003	
82	0	LOD		0	00270000	
83	0	MUL	C	1016	001483F8	CONSTANT PROPAGATION
84	0	STO		0	00280000	
85	0	DEL		0	002A0000	
86	0	ENT		114	00030072	
87	0	LOC		7	00020007	
88	0	LOC		2	00020002	
89	0	LOD		0	00270000	
90	0	STO		0	00280000	
91	0	DEL		0	002A0000	
92	0	LOC		7	00020007	
93	0	LOC		7	00020007	
94	0	LOD		0	00270000	
95	0	LOC		3	00020003	
96	0	LOD		0	00270000	
97	0	MUL	C	1000	001483E8	SIMPLIFICATION
98	0	STO		0	00280000	
99	0	DEL		0	002A0000	

\*\*\*\*\*

# FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

FINAL POOL FOR BLOCK# 1 IS:  
(NULL)

FINAL POOL FOR BLOCK# 2 IS:  
C(104:105,\*,106) A(3,100,100) A(2,\*,105) A(1,\*,104)

FINAL POOL FOR BLOCK# 3 IS:  
C(104:105,\*,106) A(3,100,100) A(2,\*,105) A(1,\*,104)

FINAL POOL FOR BLOCK# 4 IS:  
A(2,\*,111) A(3,100,100) A(1,\*,104) C(111:100,\*,111)

FINAL POOL FOR BLOCK# 5 IS:  
C(104:105,\*,106) A(3,100,100) A(2,\*,105) A(1,\*,104)

FINAL POOL FOR BLOCK# 6 IS:  
C(104:105,\*,106) A(3,100,100) A(2,\*,105) A(1,\*,104)

FINAL POOL FOR BLOCK# 7 IS:  
A(2,\*,110) A(3,100,100) A(1,\*,104) C(110:100,\*,110)

BASIC BLOCK #1  
BEGINNING AT 0, ENDING AT 19  
BLOCK TRAVERSED 1 TIMES  
0 REFERENCES:  
BASIC BLOCK #2  
BEGINNING AT 73, ENDING AT 86  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
1  
BASIC BLOCK #3  
BEGINNING AT 20, ENDING AT 28  
BLOCK TRAVERSED 1 TIMES  
1 REFERENCES:  
1  
BASIC BLOCK #4



```

      BEGINNING AT 86, ENDING AT 99
      BLOCK TRAVERSED 2 TIMES
      1 REFERENCES:
        2
BASIC BLOCK #5
      BEGINNING AT 44, ENDING AT 57
      BLOCK TRAVERSED 1 TIMES
      1 REFERENCES:
        3
BASIC BLOCK #6
      BEGINNING AT 29, ENDING AT 43
      BLOCK TRAVERSED 1 TIMES
      1 REFERENCES:
        3
BASIC BLOCK #7
      BEGINNING AT 57, ENDING AT 72
      BLOCK TRAVERSED 1 TIMES
      1 REFERENCES:
        6

```

BLOCK SUMMARY DATA:

BLOCK NUMBER	NUMBER OF PASSES
1	1
2	1
3	1
4	2
5	1
6	1
7	1

TOTAL NUMBER OF BLOCKS PROCESSED WAS 8  
 USING THE FIRST-IN-FIRST-OUT BLOCK SELECTION ALGORITHM.  
 TIME FOR THE OPTIMIZATION WAS 0.449 SECONDS.

\*\*\*CODE SYNTHESIS FILTER IS COMPLETE\*\*\*





CARD	BL	SYL	COMMENT
1	0	0	TEST PROGRAM # 4 ;
\$EXECUTE			
3	0	1	BEGIN LOCAL A,B,C,D,E,F,G,H,I,J,K;
4	1	1	READ(A, B, C);
5	1	10	I := B * C;
6	1	18	WHILE A GTR B DO
7	1	28	BEGIN
8	1	28	WHILE A GTR B DO
9	2	38	BEGIN
10	2	38	WHILE A GTR B DO
11	3	48	J := B * C;
12	3	59	K := C * B;
13	3	67	END;
14	2	70	END;
15	1	73	END
16	1	73	EOF

CODE FILE COPIED (73 WORDS)  
 CONSTANT TABLE COPIED (0 WORDS)  
 2 RECORDS WRITTEN INTO FILE 1  
 END OF COMPILATION FEBRUARY 12, 1975. CLOCK TIME = 21:13:33.31.  
 16 CARDS WERE READ.  
 NO ERRORS WERE DETECTED.



\*\*\*CODE SYNTHESIS FILTER\*\*\*

BLOCK SELECTION METHOD WILL BE MAXIMUM CURRENT POOL

\*\*\*C O M P L E T E T A B L E D U M P\*\*\*  
DUMP REQUESTED FROM BLOCK NUMBER = 0 AT LOCATION = 0

\*\*\*T A B L E D U M P\*\*\* AT LOCATION = 0

CONSTANTS <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

ADDR TABL <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

VALUE STACK (TOP AT 127)

CONT STK <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
BLOCK# 10  
ENTRY 10

EXEC STAC <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14>  
(EMPTY)

```

**** FORMATTED INTERMEDIATE CODE DUMP ****
LOC OFF  OP CODE  CNS ADR  RAW CODE  OPTIMIZATION
SET      /ETC      TYPE
*****
0      0      ENT      C 74      0003804A
1      0      TOGGLE      257      000B0101
2      0      LOC      1      00020001
3      0      INT      0      00050000
4      0      STD      0      00290000
5      0      LOC      2      00020002
6      0      INT      0      00050000
7      0      STD      0      00290000
8      0      LOC      3      00020003
9      0      INT      0      00050000
10     0      STD      0      00290000
11     0      LOC      9      00020009
12     0      LOC      2      00020002
13     0      LOD      0      00270000
14     0      LOC      3      00020003
15     0      LOD      0      00270000
16     0      MUL      0      00140000
17     0      STD      0      00280000
18     0      DEL      0      002A0000
19     0      ENT      1      00030001
20     0      LOC      1      00020001
21     0      LSD      0      00270000
22     0      LOC      2      00020002
23     0      LOD      0      00270000
24     0      GTR      0      00210000
25     0      XIT      73      00040049
26     0      XIT      28      0004001C
27     0      BSC      0      002F0000
28     0      ENT      1      00030001
29     0      ENT      1      00030001
30     0      LOC      1      00020001
31     0      LOD      0      00270000
32     0      LOC      2      00020002
33     0      LOD      0      00270000
34     0      GTR      0      00210000
35     0      XIT      70      00040046
36     0      XIT      38      00040026
37     0      BSC      0      002F0000
38     0      ENT      1      00030001
39     0      ENT      1      00030001
40     0      LOC      1      00020001
41     0      LOD      0      00270000
42     0      LOC      2      00020002
43     0      LOD      0      00270000
44     0      GTR      0      00210000

```



45	0	XIT	59	00040038
46	0	XIT	48	00040030
47	0	BSC	0	002F0000
48	0	ENT	1	00030001
49	0	LOC	10	0002000A
50	0	LOC	2	00020002
51	0	LOC	0	00270000
52	0	LOC	3	00020003
53	0	LOC	0	00270000
54	0	MUL	0	00140000
55	0	STO	0	00280000
56	0	DEL	0	002A0000
57	0	XIT	39	00040027
58	0	BRS	0	002E0000
59	0	ENT	1	00030001
60	0	LOC	11	00020008
61	0	LOC	3	00020003
62	0	LOC	0	00270000
63	0	LOC	2	00020002
64	0	LOC	0	00270000
65	0	MUL	0	00140000
66	0	STO	0	00280000
67	0	DEL	0	002A0000
68	0	XIT	29	00040010
69	0	BRS	0	002E0000
70	0	ENT	1	00030001
71	0	XIT	19	00040013
72	0	BRS	0	002E0000
73	0	ENT	1	00030001

\*\*\*\*\*

BASIC BLOCK #1  
 BEGINNING AT 0, ENDING AT 0  
 BLOCK TRAVERSED 0 TIMES  
 0 REFERENCES:

\*\*\*END OF COMPLETE TABLE DUMP\*\*\*

B+2	0	LOC	1	00020001
B+3	0	INT	0	00050000
B+4	0	STO	0	00290000
B+5	0	LOC	2	00020002
B+6	0	INT	0	00050000
B+7	0	STO	0	00290000
B+8	0	LOC	3	00020003
B+9	0	INT	0	00050000
B+10	0	STO	0	00290000
B+11	0	LOC	9	00020009
B+12	0	LOC	2	00020002
B+13	0	LOC	0	00270000
B+14	0	LOC	3	00020003
B+15	0	LOC	0	00270000
B+16	0	MUL	0	00140000
B+17	0	STO	0	00280000
B+18	0	DEL	0	002A0000
B+19	0	ENT	1	00030001

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 2 IS:  
 C(75:76,\*,77) A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74)

CONTROL AT 19, FROM BLK 1 TO BLK 2 (PASS 1)

C+19	0	ENT	C 78	0003804E
B+20	0	LOC	1	00020001
B+21	0	LOC	0	00270000
B+22	0	LOC	2	00020002
B+23	0	LOC	0	00270000
B+24	0	GTR	0	00210000
B+25	0	XIT	73	00040049
B+26	0	XIT	28	0004001C
B+27	0	BSC	0	002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 3 IS:  
 C(75:76,\*,77) C(74:75,\*,78) A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 4 IS:  
 C(75:76,\*,77) C(74:75,\*,78) A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74)

CONTROL AT 28, FROM BLK 2 TO BLK 4 (PASS 1)

C+28	0	ENT	C 88	00038058
B+29	0	ENT	1	00030001

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 5 IS:  
 C(74:75,\*,78) C(75:76,\*,77) A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74)

CONTROL AT 29, FROM BLK 4 TO BLK 5 (PASS 1)

C+29	0	ENT	C 93	0003805D
------	---	-----	------	----------



B+30	0	LUL	1	00020001
B+31	0	LOD	0	00270000
B+32	0	LOC	2	00020002
B+33	0	LOD	0	00270000
B+34	0	GTR	0	00210000
B+35	0	XIT	70	00040046
B+36	0	XIT	38	00040026
B+37	0	BSC	0	002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 6 IS:  
C(75:76,\*,77) C(74:75,\*,78) A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 7 IS:  
C(75:76,\*,77) C(74:75,\*,78) A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74)

CONTROL AT 38, FROM BLK 5 TO BLK 7 (PASS 1)				
C+38	0	ENT	C 103	00038067
B+39	0	ENT	1	00030001

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 8 IS:  
C(74:75,\*,78) C(75:76,\*,77) A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74)

CONTROL AT 39, FROM BLK 7 TO BLK 8 (PASS 1)				
C+39	0	ENT	C 108	0003806C
B+40	0	LOC	1	00020001
B+41	0	LOD	0	00270000
B+42	0	LOC	2	00020002
B+43	0	LOD	0	00270000
B+44	0	GTR	0	00210000
B+45	0	XIT	59	00040038
B+46	0	XIT	48	00040030
B+47	0	BSC	0	002F0000

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 9 IS:  
C(75:76,\*,77) C(74:75,\*,78) A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74)

INITIAL CURRENT OPTIMIZED POOL TO BE USED FOR BLOCK# 10 IS:  
C(75:76,\*,77) C(74:75,\*,78) A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74)

CONTROL AT 48, FROM BLK 8 TO BLK 10 (PASS 1)				
C+48	0	ENT	C 118	00038076
B+49	0	LOC	10	0002000A
B+50	0	LOC	2	00020002
B+51	0	LOD	0	00270000
B+52	0	LOC	3	00020003
B+53	0	LOD	0	00270000
B+54	0	MUL	0	00140000
B+55	0	STD	0	00280000
B+56	0	DEL	0	002A0000
B+57	0	XIT	39	00040027
B+58	0	BRS	0	002E0000

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 10 IS:  
C(74:75,\*,78) C(75:76,\*,77) A(10,\*,77) A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74)

CURRENT POOL FOR BLOCK# 8 IS:  
C(74:75,\*,78) C(75:76,\*,77) A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74)

AFTER PERFORMING THE MEET OPERATION:  
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

CONTROL AT 59, FROM BLK 8 TO BLK 9 (PASS 1)				
C+59	0	ENT	C 113	00038071
B+60	0	LOC	11	0002000B
B+61	0	LOC	3	00020003
B+62	0	LOD	0	00270000
B+63	0	LOC	2	00020002
B+64	0	LOD	0	00270000
B+65	0	MUL	0	00140000
B+66	0	STD	0	00280000
B+67	0	DEL	0	002A0000
B+68	0	XIT	29	0004001D
B+69	0	BRS	0	002E0000

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 9 IS:  
C(74:75,\*,78) C(75:76,\*,77) A(11,\*,77) A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74)

CURRENT POOL FOR BLOCK# 5 IS: . . . . .





C(14:15,\*,78) C(15:16,\*,77) A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74)

AFTER PERFORMING THE MEET OPERATION:  
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

CONTROL AT 70, FROM BLK 5 TO BLK 6 (PASS 1)  
C+70 0 ENT C 98 00038062  
B+71 0 XIT 19 00040013  
B+72 0 BRS 0 002E0000

BEFORE THE MEET OPERATION:

OUTPUT POOL FROM BLOCK# 6 IS:  
C(74:75,\*,78) C(75:76,\*,77) A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74)

CURRENT POOL FOR BLOCK# 2 IS:  
C(75:76,\*,77) A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74)

AFTER PERFORMING THE MEET OPERATION:  
NEW CURRENT POOL/INPUT POOL IS THE SAME AS OLD CURRENT POOL

CONTROL AT 73, FROM BLK 2 TO BLK 3 (PASS 1)  
C+73 0 ENT C 83 00038053

### FINAL OPTIMIZATION RESULTS

OPTIMIZATION AT 34  
(74,75|24,78) COMM SUBEXP ELIM

OPTIMIZATION AT 44  
(74,75|24,78) COMM SUBEXP ELIM

OPTIMIZATION AT 54  
(75,76|16,77) COMM SUBEXP ELIM

OPTIMIZATION AT 65  
(75,76|16,77) COMM SUBEXP ELIM

\*\*\*\* FORMATTED INTERMEDIATE CODE DUMP \*\*\*\*  
LOC OFF OP CODE CNS ADR RAW CODE OPTIMIZATION  
SET /ETC TYPE

```

*****
0 0 ENT 74 0003004A
1 0 TOGGLE 257 000B0101
2 0 LOC 1 00020001
3 0 INT 0 00050000
4 0 STD 0 00290000
5 0 LOC 2 00020002
6 0 INT 0 00050000
7 0 STD 0 00290000
8 0 LOC 3 00020003
9 0 INT 0 00050000
10 0 STD 0 00290000
11 0 LOC 9 00020009
12 0 LOC 2 00020002
13 0 LOD 0 00270000
14 0 LOC 3 00020003
15 0 LOD 0 00270000
16 0 MUL 1024 00140400
17 0 STD 0 00280000
18 0 DEL 0 002A0000
19 0 ENT 78 0003004E
20 0 LOC 1 00020001
21 0 LOD 0 00270000
22 0 LOC 2 00020002
23 0 LOD 0 00270000
24 0 GTR 1018 002103FA
25 0 XIT 73 00040049
26 0 XIT 28 0004001C
27 0 BSC 0 002F0000
28 0 ENT 88 00030058
29 0 ENT 93 0003005D
30 0 LOC 1 00020001
31 0 LOD 0 00270000
32 0 LOC 2 00020002
33 0 LOD 0 00270000
34 0 GTR C 1002 002183EA COMM SUBEXP ELIM
35 0 XIT 70 00040046
36 0 XIT 38 00040026
37 0 BSC 0 002F0000
38 0 ENT 103 00030067
39 0 ENT 108 0003006C
*****

```



40	U	LUL	1	00020001	
41	0	LOD	0	00270000	
42	0	LOC	2	00020002	
43	0	LOD	0	00270000	
44	0	GTR	C 986	002183DA	COMM SUBEXP ELIM
45	0	XIT	59	00040038	
46	0	XIT	48	00040030	
47	0	BSC	0	002F0000	
48	0	ENT	118	00030076	
49	0	LOC	10	0002000A	
50	0	LOC	2	00020002	
51	0	LOD	0	00270000	
52	0	LOC	3	00020003	
53	0	LOD	0	00270000	
54	0	MUL	C 974	001483CE	COMM SUBEXP ELIM
55	0	STO	0	00280000	
56	0	DEL	0	002A0000	
57	0	XIT	39	00040027	
58	0	BRS	0	002E0000	
59	0	ENT	113	00030071	
60	0	LOC	11	00020008	
61	0	LOC	3	00020003	
62	0	LOD	0	00270000	
63	0	LOC	2	00020002	
64	0	LOD	0	00270000	
65	0	MUL	C 970	001483CA	COMM SUBEXP ELIM
66	0	STO	0	00280000	
67	0	DEL	0	002A0000	
68	0	XIT	29	0004001D	
69	0	BRS	0	002E0000	
70	0	ENT	98	00030062	
71	0	XIT	19	00040013	
72	0	BRS	0	002E0000	
73	0	ENT	83	00030053	

\*\*\*\*\*

# FINAL OPTIMIZATION POOL FOR EACH BLOCK IS:

FINAL POCL FOR BLOCK# 1 IS:  
(NULL)

FINAL POCL FOR BLOCK# 2 IS:  
A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74) C(75:76,\*,77)

FINAL POCL FOR BLOCK# 3 IS:  
C(75:76,\*,77) C(74:75,\*,78) A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74)

FINAL POCL FOR BLOCK# 4 IS:  
C(75:76,\*,77) C(74:75,\*,78) A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74)

FINAL POCL FOR BLOCK# 5 IS:  
A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74) C(74:75,\*,78) C(75:76,\*,77)

FINAL POCL FOR BLOCK# 6 IS:  
C(75:76,\*,77) C(74:75,\*,78) A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74)

FINAL POCL FOR BLOCK# 7 IS:  
C(75:76,\*,77) C(74:75,\*,78) A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74)

FINAL POCL FOR BLOCK# 8 IS:  
A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74) C(74:75,\*,78) C(75:76,\*,77)

FINAL POCL FOR BLOCK# 9 IS:  
C(75:76,\*,77) C(74:75,\*,78) A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74)

FINAL POCL FOR BLOCK# 10 IS:



FINAL POOL FOR BLOCK# 10 IS:  
 C(75:76,\*,77) C(74:75,\*,78) A(9,\*,77) A(3,\*,76) A(2,\*,75) A(1,\*,74)

```

BASIC BLOCK #1
BEGINNING AT 0, ENDING AT 19
BLOCK TRAVERSED 1 TIMES
0 REFERENCES:
BASIC BLOCK #2
BEGINNING AT 19, ENDING AT 27
BLOCK TRAVERSED 1 TIMES
1 REFERENCES:
1
BASIC BLOCK #3
BEGINNING AT 73, ENDING AT 73
BLOCK TRAVERSED 1 TIMES
1 REFERENCES:
2
BASIC BLOCK #4
BEGINNING AT 28, ENDING AT 29
BLOCK TRAVERSED 1 TIMES
1 REFERENCES:
2
BASIC BLOCK #5
BEGINNING AT 29, ENDING AT 37
BLOCK TRAVERSED 1 TIMES
1 REFERENCES:
4
BASIC BLOCK #6
BEGINNING AT 70, ENDING AT 72
BLOCK TRAVERSED 1 TIMES
1 REFERENCES:
5
BASIC BLOCK #7
BEGINNING AT 38, ENDING AT 39
BLOCK TRAVERSED 1 TIMES
1 REFERENCES:
5
BASIC BLOCK #8
BEGINNING AT 39, ENDING AT 47
BLOCK TRAVERSED 1 TIMES
1 REFERENCES:
7
BASIC BLOCK #9
BEGINNING AT 59, ENDING AT 69
BLOCK TRAVERSED 1 TIMES
1 REFERENCES:
8
BASIC BLOCK #10
BEGINNING AT 48, ENDING AT 58
BLOCK TRAVERSED 1 TIMES
1 REFERENCES:
8

```

# BLOCK SUMMARY DATA:

BLOCK NUMBER	NUMBER OF PASSES
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1

TOTAL NUMBER OF BLOCKS PROCESSED WAS 10  
 USING THE MAXIMUM CURRENT POOL BLOCK SELECTION ALGORITHM.  
 TIME FOR THE OPTIMIZATION WAS 0.185 SECONDS.

\*\*\*CODE SYNTHESIS FILTER IS COMPLETE\*\*\*





## COMPUTER PROGRAM

```

/*<><><><><><><><><><><><><><><>*/
/*CODE SYNTHESIS FILTER*/
/*THIS PROGRAM OPERATES ON AN INTERMEDIATE*/
/*LANGUAGE RECEIVED FROM THE ALGOL-E*/
/*COMPILER IN THE FOLLOWING FORM*/
/*BITS      8      8      1      15*/
/*OFFSET OPCODE C-FIELD ADDRESS*/
/*      / TYPE*/
/*USING THE META - EXECUTION TECHNIQUE,*/
/*THE ALGORITHM (Q) OF KILDALL IS*/
/*APPLIED TO THE INTERMEDIATE LANGUAGE.*/
/*NO CODE OPTIMIZATION IS IN FACT DONE;*/
/*RATHER UPON RECOGNITION OF A POSSIBLE*/
/*CODE INEFFICIENCY BIT 16 IN THE 32 BIT*/
/*INTERMEDIATE LANGUAGE IS SET TO INDICATE*/
/*THAT DURING CODE GENERATION AN*/
/*OPTIMIZATION CAN BE ACCOMPLISHED.*/
/*IMPORTANT DATA DESCRIPTOR DEFINITIONS:*/
/*EXEC = EXECUTION STACK FOR THE*/
/*      META EXECUTION*/
/*CSFCODE = BUFFER FOR ALGOL-E CODE*/
/*ADDRESS = VARIABLE DEFINITION TABLES*/
/*CON = CONSTANT TABLES*/
/*CONTROL = THE LIST OF BLOCKS TO BE*/
/*           PROCESSED...PGM TERMINATES*/
/*           WHEN THIS TABLE IS PAU*/
/*STATE = POINTER TABLE TO CURRENT*/
/*        STATE INFO FOR EACH BLOCK*/
/*        PROCESSED*/
/*VTAB = EXPRESSION STACK FOR ALL*/
/*       EXPRESSIONS*/
/*EX = EXECUTION STACK FOR THE*/
/*     META-EXECUTIONS*/
/*OPTIM_TYPE = CODE FOR THE OPTIMIZATION*/
/*             TYPE DETECTED*/
/*<><><><><><><><><><><><><><><>*/
/*THE FOLLOWING TABLES ARE OUTPUT FROM THE*/
/*PROGRAM "GENERATOR" AND ARE USED TO*/
/*DESCRIBE THE INTERMEDIATE LANGUAGE.*/
/*<><><><><><><><><><><><><><><>*/
/*BASIC OPERATORS SET TO 0,1,ETC IN INITIAL*/
/*<><><><><><><><><><><><><><><>*/
DECLARE(NULL, DEFAULT, LOC, ENT, XIT, INT, REAL, BOOL, IARRAY, RARRAY) BIT(8),
NTYPES LITERALLY'9',
/*<><><><><><><><><><><><><><><>*/
/*REMAINDER OF THE OPERATIONS CODES*/
/*<><><><><><><><><><><><><><><>*/
/*(REFER, TOGGLE, PASS, TRU, RND, FLT, ADD, RADD, SUB, RSUB, MUL, RMUL, DIV,
RDIV, EXP, REXP, IRXP, RRXP, LSS, LEQ, EQL, NEQ, GEQ, GTR, INEG, RNEG, NDT,
AND, BOR, LOD, STD, DEL, DUP, XCH, HLT, BRS, BSC, NOP, PRO, RTN, GET, RET,
RRD, IRD, BRD, WRV, DMP, TAB, SUP)BIT(8),
NOPCODES LITERALLY'50',
/*<><><><><><><><><><><><><><><>*/
/*CHARACTER STRINGS OF LEGAL OPCODES -
OPRANGE IS USED TO ACCESS
/*<><><><><><><><><><><><><><><>*/
/*OPCODES CHARACTER INITIALI
'NULLDEFAULTLOCENTXITINTREALBOOLIARRAYRARRAYREFERTOGGLEPASSTRURNDFLTADDRADDSUBRS
UBMULRMULDIVRDIEXPRIXPIRRXPRLSSLEGECLNEQEQGTKINEGRNEGNOTANDBORLOOSTOSTDDELDO
PXCHHLTBRSBSCNOPPRKTNGETKRTDRDIRDBRWVDMPTABSUP'),

```















```

251 RESIND FIXED,
252
253 OPERTR FIXED,
254 HASHADR FIXED,
255 MAXADD LITERALLY '850',
256 ADDRESS(MAXADD) FIXED,
257 ADDTYPE(MAXADD) BIT(8),
258 ADDVAL(MAXADD) BIT(16),
259 ADDCON(MAXADD) BIT(16),
260 ADDLOC FIXED,
261 ADDNAM(MAXADD) BIT(16),
262
263
264 MAXCON LITERALLY '50',
265 CONSYM(MAXCON) CHARACTER,
266 CONTYPE(MAXCON) BIT(8),
267 CONADD(MAXCON) BIT(16),
268 CONVAL(MAXCON) BIT(16),
269 CONINT(MAXCON) FIXED,
270 CONLOC FIXED,
271 PASS# FIXED,
272 #XITS FIXED,
273 VALUE_NUM_CNTR FIXED,
274
275 CSF_FILE LITERALLY '1',
276
277
278 DISKWORDS LITERALLY '900',
279 MAXCODE LITERALLY '2700',
280 #PARMS LITERALLY '0',
281 CODELOC FIXED,
282 CSFCODE(MAXCODE) FIXED,
283 OPTIM_TYPE(MAXCODE) BIT(B),
284
285 LASTCON FIXED,
286 MAXCONT LITERALLY '50',
287 CONTROL(MAXCONT) FIXED,
288
289
290 SAV_TOG(MAXCONT) BIT(8),
291 SELECT_FLAG FIXED,
292
293
294 MAXBLK LITERALLY '90',
295
296 BLK# FIXED,
297 BLKCNT FIXED,
298 BLKLLOC FIXED,
299
300 BLKPNT(MAXBLK) FIXED,
301
302 BLKSIZE LITERALLY '4',
303
304 M_CONTROL FIXED,
305 CTOP FIXED,
306 CURBLOCK FIXED,
307
308 MAX_OPTIMIZATIONS LITERALLY '100',
309
310 OPTIM_ADR_LIST(MAX_OPTIMIZATIONS) FIXED,
311
312
313
314
315 OPTOP FIXED,
316
317
318
319
320
321
322
323 FIRSTCALL BIT(1),
324
325 BOPA FIXED,
326 BOPB FIXED,
327 TADDLOC FIXED,
328
329
330 NUM_SELECT_METHODS LITERALLY '3',
331
332 STRTIME FIXED,
333
334 PAUTIME FIXED,
335
336 TOTIME FIXED,
337
338 SECONDS FIXED,
339 PRTSEC FIXED,
340

```

```

/* INDICATOR FOR WHERE RESULT */
/* VALUE NUMBER IS FOUND */
/* OPERATOR */
/* COLLISION HASH ADDRESS */
/* MAX LIMIT OF ADDRESS TABLE */
/* INTERNAL ADDR NUMBER */
/* TYPE OF NUMBER */
/* EQUAL. CLASS NUMBER */
/* POINT TO CONSTANT TAB */
/* POINT TO ADDRESS TABLE */
/* IF ADDRESS ENTRY IS EQUAL */
/* TO A COMPUTATION, THIS IS */
/* THE POINTER TO VTAB */
/* LIMIT FOR CONSTANTS */
/* INTERVAL SYMBOL REPRESENTATI */
/* TYPE OF CONSTANT..INT,ETC */
/* CONSTANT ADDRESS */
/* EQUAL. CLASS NUMBER */
/* CONSTANT INTERNAL VALUE */
/* POINTER TO CONSTANT TABS */
/* PASSES FOR CURRENT BLOCK */
/* EXITS CURRENT BLOCK */
/* ACTUAL COUNTER USED TO */
/* GENERATE VALUE NUMBERS */
/* FILE # UPON WHICH THE */
/* INTERMEDIATE LANGUAGE CAN */
/* BE FOUND */
/* 3600 BYTES */
/* MAX CODE ALLOWED */
/*
/* POINTER TO CSFCODE */
/* INPUT CODE/CONS AREA */
/* TABLE TO DECLARE TYPE OF */
/* OPTIMIZATION FOUND: */
/* POINT TO THE LAST CONSTANT */
/* LIMIT FOR CONTROL TABLE */
/* BLK# - BLK FWA ** THIS IS */
/* THE LIST FOR THE TABLE */
/* PROCESSED BLOCKS. */
/* TOGGLES FOR THIS BLOCK */
/* SET TO INDICATE TYPE OF */
/* SELECTION METHOD DESIRED */
/* FROM CONTROL LIST: */
/* MAXIMUM # OF BLOCKS IN A */
/* PROGRAM */
/* HOLDS CURRENT BLK# */
/* # OF BLOCKS */
/* POINTS TO LWA + 1 OF THE */
/* BLOCK HEADER AREA */
/* HOLDS FWA IN CSFCODE FOR */
/* EACH BLOCK HEADER */
/* PRESET MINIMUM SIZE OF */
/* BLOCK HEADER */
/* POINT TO CSF FILE FOR FLOW */
/* CONTROL STACK POINTER */
/* HOLDS ADDRESS OF CURRENT */
/* BLOCK HEADER */
/* LIMIT ON THE NUMBER OF */
/* OPTIMIZATIONS THAT CAN BE */
/* FOUND. */
/* HOLDS THE OPTIMIZATION EXP */
/* BLOCK ADDRESS IN THE TOP */
/* OF VTAB FOR EVERY POSSIBLE */
/* OPTIMIZATION. */
/* POINTER TO OPTIM_ADR_LIST */
/*
/* USED IN PERMUTE TO INDICATE */
/* THE FIRST PASS FOR A STRING */
/* FIRST OPERAND POINTER */
/* SECOND OPERAND POINTER */
/* CURRENT BOTTOM OF ADDRESS */
/* TABLE FOR BLOCK BEING */
/* PROCESSED */
/* MAX NUMBER FOR BLOCK SELECT */
/* CODE */
/* HOLDS THE TIME FOR START OF */
/* OPTIM */
/* HOLDS COMPLETION TIME FOR */
/* OPTIM */
/* HOLDS TOTAL TIME FOR OPTIM */
/* IN CENTISECONDS */
/* HOLDS SECONDS FOR OPTIM */
/* HOLDS PARTIAL SECONDS FOR */
/* OPTIM

```









[illegible]







[illegible]





```

687      /*<><><><><><><><><><><><><><><><><><>*/
688      /* W_EXEC */
689      /* PRINT THE EXEC STACK TABLES */
690      /* ROUTINES CALLED: */
691      /* W_TITLE */
692      /* LINE_FEED */
693      /* W_CON */
694      /* W_TYPE */
695      /* W_NUM */
696      /* W_CLEAR */
697      /*<><><><><><><><><><><><><><><><><><>*/
698 W_EXEC: PROCEDURE;
699   DECLARE (I,J) FIXED;
700   ALTEX=0;
701   CALL W_TITLE(' EXEC STACK');
702   IF ETOP < 1 THEN
703     DO;
704       OUTPUT = ' (EMPTY)';
705       IF PUNCHTOG THEN
706         OUTPUT(2) = ' (EMPTY)';
707       CALL LINE_FEED(2);
708       RETURN;
709     END;
710   DO I = 0 TO #EXF;
711     HEADING=EXHEAD(I);
712     DO J = 1 TO ETOP;
713       DO CASE I;
714         CALL W_CON(J,EXCON(J));
715         CALL W_TYPE(J,EXTYPE(J));
716         CALL W_NUM(J,EXADD(J));
717         CALL W_NUM(J,EXVAL#(J));
718       END;
719     END;
720     CALL W_CLEAR;
721   END;
722   CALL LINE_FEED(2);
723 END W_EXEC;
724      /*<><><><><><><><><><><><><><><><><><>*/
725      /* W_ADD */
726      /* PRINTS THE ADDRESS TABLES. */
727      /* ROUTINES CALLED: */
728      /* W_TITLE */
729      /* LINE_FEED */
730      /* W_NUM */
731      /* W_TYPE */
732      /* W_CLEAR */
733      /*<><><><><><><><><><><><><><><><><><>*/
734 W_ADD: PROCEDURE;
735   DECLARE (I, J) FIXED;
736   ALTAD=0;
737   CALL W_TITLE(' ADDR TABLE');
738   IF ADDLOC < 1 THEN
739     DO;
740       OUTPUT = ' (EMPTY)';
741       IF PUNCHTOG THEN
742         OUTPUT(2) = ' (EMPTY)';
743       CALL LINE_FEED(2);
744       RETURN;
745     END;
746   DO I = 0 TO #ADF;
747     HEADING=ADHEAD(I);
748     DO J = 1 TO ADDLOC;
749       DO CASE I;
750         CALL W_NUM(J,ADDRESS(J));
751         CALL W_TYPE(J,ADDTYPE(J));
752         CALL W_NUM(J,ADDVAL(J));
753         CALL W_NUM(J,ADDCON(J));
754       END;
755     END;
756     CALL W_CLEAR;
757   END;
758   CALL LINE_FEED(2);
759 END W_ADD;
```









```

839      /*<><><><><><><><><><><><><><><><><><>*/
840      /*
841      /*          W_VAL
842      /* OUTPUTS THE VALUE STACK IN THE FORMAT
843      /*          C(X Y Z | W)
844      /*          X = OPERATOR
845      /*          Y = CLASS # OPERAND A
846      /*          Z = CLASS # OPERAND B
847      /*          W = CLASS # FOR COMPUTATION
848      /* ROUTINES CALLED:
849      /*          W_CLEAR
850      /*          LINE_FEED
851      /*
852      /*<><><><><><><><><><><><><><><><><>*/
853      W_VAL: PROCEDURE;
854      DECLARE (I,J,N,K,L,M) FIXED , T CHARACTER;
855      ALTVL=0;
856      TBUF = ' ';
857      N = T_WIDTH + N_COLS * C_WIDTH;
858      OUTPUT = ' VALUE STACK (TOP AT '|VTOP|'|)';
859      IF PUNCHTOG THEN
860      OUTPUT(2) = ' VALUE STACK (TOP AT '|VTOP|'|)';
861      I = VTOP;
862      J = VTAB(VTOP);
863      DO WHILE J > 0;
864      K = VTAB(J + 1);
865      IF SHR(K, 15) THEN T = 'F';
866      ELSE T = 'C';
867      M = K & "FF";
868      L = OPRANGE(M);
869      K = OPRANGE(M + 1) - L;
870      T=T||'('||SUBSTR(OPCODES,L,K);
871      K = OP_DEGL(M);
872      L = J + 3;
873      DO WHILE K > 0;
874      K = K - 1;
875      T = T || ' ' || VTAB(L);
876      L = L + 1;
877      END;
878      T=T||') '|VTAB(L)||' ';
879      I = J;
880      J = VTAB(J);
881      IF LENGTH(TBUF) + LENGTH(T) > N THEN
882      DO;
883      OUTPUT = TBUF;
884      IF PUNCHTOG THEN
885      OUTPUT(2) = TBUF;
886      TBUF = T;
887      END;
888      ELSE TBUF = TBUF || T;
889      END;
890      CALL W_CLEAR;
891      CALL LINE_FEED(2);
892      END W_VAL;
893      /*<><><><><><><><><><><><><><><><><>*/
894      /*
895      /*          W_STATE
896      /* OUTPUTS THE STATE CORRESPONDING TO
897      /* BLOCK AT ADDRESS CB IN CSFCODE.
898      /* ROUTINES CALLED:
899      /*          LINE_FEED
900      /*          W_CLEAR
901      /*
902      /*<><><><><><><><><><><><><><><><><>*/
903      W_STATE: PROCEDURE (CB, WHERE);
904      DECLARE (I, K, J, M, CB, X, TEMP, NUMOP) FIXED,
905      (T, WHERE) CHARACTER;
906      CALL LINE_FEED(1);
907      ALTVL = 0;
908      IF CB = 0 THEN RETURN;
909      M=T_WIDTH+N_COLS*C_WIDTH;
910      I = CSFCODETCB;
911      K = SHR(I,16);
912      I = I & "FFFF";
913      TBUF = ' ';
914      OUTPUT = ' ' || WHERE || ' BLOCK# ' || K || ' IS: ';
915      IF PUNCHTOG THEN
916      OUTPUT(2) = ' ' || WHERE || ' BLOCK# ' || K || ' IS: ';
917      CB=STATES(J+1);
918
919      /*
920      /* THERE ARE TWO EMPTY SPACES
921      /* AT THE BEGINNING OF THE
922      /* STATE VECTOR.
923
924      I = I + 2;
925      J = STATES(I);
926
927      /*
928      /* IS AN OPTIMIZED STATE
929      /* PRESENT

```



```

929 IF J = 0 THEN
930
931
932
933 DO;
934   OUTPUT = ' (NULL)';
935   IF PUNCHTOG THEN
936     OUTPUT(2) = ' (NULL)';
937 END;
938 ELSE DO;
939
940
941
942
943 DO WHILE J > 0;
944   J = J - 1;
945   I = I + 1;
946   K = STATES(I) & "7FFF";
947   X = ADDRESS(K);
948   IF SHR(X, 15) THEN
949
950
951
952
953
954
955
956
957
958
959   DO;
960     T = ' E(';
961     X = X & "7FFF";
962   END; ELSE
963     IF X = 0 THEN
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
END W_CLEAR;
CALL LINE_FEED(1);
END W_STATE;

```

```

/* NO...INDICATE EMPTY */
/* */

```

```

/* YES...PROCESS EACH */
/* EQUIVALENCE CLASS */
/* */

```

```

/* EXEC STACK ELEMENT */
/* E ( X, Y, Z ) */
/* X = RELATIVE LOCATION */
/* FROM THE TOP OF */
/* EXEC STACK */
/* Y = IF RESULT IS */
/* PROPAGATED CONSTANT */
/* VALUE IS CLASS # */
/* Z = CLASS NUMBER OF THE */
/* COMPUTATION RESULT */

```

```

/* VALUE TABLE ENTRY */
/* INVOLVING AT LEAST ONE */
/* COMPUTATION. */
/* C ( X,Y,Z ) */

```

```

/* OPERANDS */
/* Y = VALUE# OF THE */
/* PROPAGATED CONSTANT */
/* IF ONE EXISTS */
/* Z = CLASS NUMBER OF */
/* COMPUTATION RESULT */

```

```

/* ADDRESS TABLE ENTRY */
/* A ( X, Y, Z ) */
/* X = SIMPLE ID NUMBER */
/* Y = SAME AS ABOVE */
/* Z = SAME AS ABOVE */

```



```

1022      /*<><><><><><><><><><>*/  

1023      /**  

1024      /* PRINT-OUTPUT POOL  

1025      /* PRINTS THE OUTPUT POOL FROM THE CURRENT  

1026      /* DATA ON THE ADDRESS STACK IN THE SAME  

1027      /* MANNER AS W_STATE PRINTS THE CURRENT  

1028      /* POOL.  

1029      /* ROUTINES CALLED:  

1030      /* ERROR  

1031      /* W_CLEAR  

1032      /* LINE_FEED  

1033      /*><><><><><><><><><><><><><><>*/  

1034      PRINT_OUTPUT_POOL: PROCEDURE;  

1035      DECLARE (CURADR, IDENTOP, ID, DATA, NUMOP, TEMP, MAXWIDTH, I,  

1036              TEMPA) FIXED,  

1037              MAXIDENTS LITERALLY '50',  

1038              IDENT(MAXIDENTS) FIXED,  

1039              T CHARACTER,  

1040              (PRESENT, EXPRES, FOUND) BIT(1);  

1041      MAXWIDTH = T_WIDTH + N_COLS * C_WIDTH;  

1042      CURADR = ADDLOC;  

1043      TBUF = ' ';  

1044      OUTPUT = ' OUTPUT POOL FROM BLOCK# '||BLK#|| ' IS: '  

1045      IF PUNCHTOG THEN  

1046      OUTPUT(2) = ' OUTPUT POOL FROM BLOCK# '||BLK#|| ' IS: '  

1047      IF CURADR < 1 THEN  

1048      DO;  

1049          OUTPUT = ' (NULL)';  

1050          IF PUNCHTOG THEN  

1051          OUTPUT(2) = ' (NULL)';  

1052          RETURN;  

1053      END;  

1054      IDENTOP = 1;  

1055      EXPRES = FALSE;  

1056      DO WHILE CURADR > 0;  

1057          FOUND = FALSE;  

1058          DATA, ID = ADDRESS(CURADR);  

1059          IF SHR(ID, 15) THEN                                /* EXEC STACK ENTRY */  

1060          DO;  

1061              FOUND = TRUE;  

1062              T = ' E(';  

1063              DATA = DATA & "7FFF";  

1064          END;  

1065          IF ID = 0 | EXPRES THEN  

1066          DO;  

1067              FOUND = TRUE;  

1068              IF EXPRES THEN  

1069              TEMP = VTAB(TEMPA) + 2;  

1070                                  /*  

1071                                  /* EXPRESSION ENTRY  

1072                                  /*  

1073              ELSE  

1074              TEMP = ADDNAM(CURADR) + 2;  

1075              TEMPA = TEMP - 1;  

1076              IF VTAB(TEMPA) /= 0 THEN  

1077              EXPRES = TRUE;  

1078              ELSE  

1079              EXPRES = FALSE;  

1080              T = ' C(';  

1081              NUMOP = ADDTYPE(CURADR) & "FF";  

1082              NUMOP = OP_DEGL(NUMOP) - 1;  

1083              DO WHILE NUMOP > 0;  

1084                  NUMOP = NUMOP - 1;  

1085                  DATA = VTAB(TEMP);  

1086                  T = T || DATA || ':';  

1087                  TEMP = TEMP + 1;  

1088              END;  

1089              DATA = VTAB(TEMP);  

1090              ELSE  

1091              DO;                                     /* ADDRESS STACK ENTRY */  

1092                  PRESENT = FALSE;  

1093                  I = IDENTOP - 1;  

1094                  DO WHILE (I > 0) & (~PRESENT);  

1095                      IF IDENT(I) = DATA THEN  

1096                      PRESENT = TRUE;  

1097                      I = I - 1;  

1098                  END;  

1099                  IF ~PRESENT THEN  

1100                  DO;  

1101                      IDENT(IDENTOP) = DATA;  

1102                      IDENTOP = IDENTOP + 1;  

1103                      IF IDENTOP > MAXIDENTS THEN  

1104                      DO;  

1105                          CALL ERROR('IDENTIFIER TABLE OVERFLOW', 1);  

1106                          IDENTOP = IDENTOP - 1;  

1107                      END;  

1108                      FOUND = TRUE;  

1109                      T = ' A(';  

1110                  END;  


```













[illegible]





[illegible]





```

1463      OUTPUT = '
1464      OUTPUT = '
1465
1466      OUTPUT = '#####'
1467      #####
1468      OUTPUT = '
1469
1470      IF PUNCHTOG THEN
1471      DO;
1472          OUTPUT(2) = '
1473      ;
1474          OUTPUT(2) = '
1475
1476      OUTPUT(2) = '#####'
1477      #####
1478      OUTPUT(2) = '
1479
1480      END;
1481      CALL LINE_FEED(1);
1482      FOUND = FALSE;
1483      DO I = #PARMS TO CODELOC;
1484          J = CSFCODE(I);
1485          K = SHR(J, 16) & "FF";
1486          D = OP_TYPE(K);
1487
1488          /* IF OPCODE IS NOT A
1489          /* DECLARATION OR A LOAD
1490          /* TEST FOR AN OPTIMIZATION
1491          /*
1492          IF (K>PASS)&(D<=LOAD) THEN
1493          DO;
1494              D = OP_DEGL(K);
1495              K = J & "7FFF";
1496              J = SHR(J, 15) & "1";
1497              J = J & (OPTIM_TYPE(I) <= "0");
1498
1499              /* IF BIT 16 IS SET, AN
1500              /* OPTIMIZATION TYPE IS
1501              /* PRESENT AND AN ADDRESS FOR
1502              /* THE OPTIMIZATION EXPRESSION
1503              /* IS PRESENT--WE HAVE AN
1504              /* OPTIMIZATION.
1505              /*
1506              IF (K<=0)&J THEN
1507              DO;
1508                  FOUND = TRUE;
1509                  J = D;
1510                  OUTPUT = ' OPTIMIZATION AT '||I;
1511                  IF PUNCHTOG THEN
1512                      OUTPUT(2) = ' OPTIMIZATION AT '||I;
1513                  T = '
1514                  DO WHILE (D > 0);
1515                      D = D - 1;
1516                      T = T || VTAB(K - D);
1517                      IF D > 0 THEN T = T || ',';
1518                  END;
1519                  T = T || '||';
1520                  D = VTAB(K - J);
1521                  IF D = 0 THEN T = T || '*,';
1522                  ELSE T = T || D || '*,';
1523                  D = VTAB(K - J - 1);
1524                  IF D = 0 THEN T = T || '*,';
1525                  ELSE T = T || D || '*,';
1526                  T = PAD(T, 30, RIGHT);
1527                  E = OPTIM_TYPE(I);
1528                  OPTYP = TYPOPTIM(E);
1529                  T = T || OPTYP;
1530                  OUTPUT = T;
1531                  IF PUNCHTOG THEN
1532                      OUTPUT(2) = T;
1533                  CALL LINE_FEED(1);
1534              END;
1535          END;
1536          IF OPTIM_TYPE(I) = "0" THEN
1537              CSFCODE(I) = CSFCODE(I) & "FFFF7FFF";
1538      END;
1539      IF ~FOUND THEN
1540      DO;
1541          OUTPUT = ' ***NO OPTIMIZATIONS DETECTED***';
1542          IF PUNCHTOG THEN
1543              OUTPUT(2) = ' ***NO OPTIMIZATIONS DETECTED***';
1544      END;
1545      CALL FORMATCODEDUMP;
1546
1547      /* OUTPUT THE MINIMAL STATE
1548      /* AT EACH BLOCK
1549      /*
1550      CALL LINE_FEED(2);

```



















```

1812      /*<><><><><><><><><><><><><><><>*/
1813      /*                                     */
1814      /*          ENTERADD                      */
1815      /* ENTERS AN IDENTIFIER INTO THE ADDRESS */
1816      /* TABLES.                             */
1817      /* ROUTINES CALLED:                     */
1818      /*          ERROR                       */
1819      /*                                     */
1820      /*<><><><><><><><><><><><><><><>*/
1821      ENTERADD: PROCEDURE(ADR);
1822      DECLARE ADR FIXED;
1823      ADDLOC = ADDLOC + 1;
1824      ALTAO=1;
1825      IF ADDLOC > ADDLOCR THEN
1826      DO;
1827          CALL ERROR('ADDRESS TABLE OVERFLOW', 1);
1828          ADDLOC = 1;
1829      END;
1830      ADDRESS(ADDLOC) = ADR;
1831      END ENTERADD;
1832      /*<><><><><><><><><><><><><><><>*/
1833      /*                                     */
1834      /*          LOOKCONS                      */
1835      /* LOOKS FOR A CHARACTER STRING AND TYPE */
1836      /* IN THE CONSTANT TABLES.           */
1837      /* RETURNS 0 IF NOT FOUND...           */
1838      /* RETURNS THE LOCATION IF FOUND....   */
1839      /*                                     */
1840      /*<><><><><><><><><><><><><><><>*/
1841      LOCKCONS: PROCEDURE(SYMSTRG, TYP) FIXED;
1842      DECLARE SYMSTRG CHARACTER, (FNDADR, TYP, CURADR) FIXED;
1843      CURADR = CONLOC;
1844      FNDADR = 0;
1845      DO WHILE (CURADR > 0) & (FNDADR = 0);
1846          IF (SYMSTRG = CONSYM(CURADR)) & (CONTYPE(CURADR) = TYP) THEN
1847              FNDADR = CURADR;
1848              CURADR = CURADR - 1;
1849      END;
1850      RETURN FNDADR;
1851      END LOCKCONS;
1852      /*<><><><><><><><><><><><><><><>*/
1853      /*                                     */
1854      /*          LOOKCONI                      */
1855      /* CHECKS FOR A CONSTANT VALUE AND TYPE IN */
1856      /* THE CONSTANT TABLE.                 */
1857      /* TRUE = RETURNS THE LOCATION FOUND     */
1858      /* FALSE = RETURNS 0                    */
1859      /*                                     */
1860      /*<><><><><><><><><><><><><><><>*/
1861      LOOKCONI: PROCEDURE (VAL#, TYP) FIXED;
1862      DECLARE (VAL#, TYP, CURADR, FNDADR) FIXED;
1863      CURADR = CONLOC;
1864      FNDADR = 0;
1865      DO WHILE (CURADR > 0) & (FNDADR = 0);
1866          IF VAL# = CONINT(CURADR) THEN
1867              IF TYP = CONTYPE(CURADR) THEN
1868                  FNDADR = CURADR;
1869                  CURADR = CURADR - 1;
1870      END;
1871      RETURN FNDADR;
1872      END LOOKCONI;
1873      /*<><><><><><><><><><><><><><><>*/
1874      /*                                     */
1875      /*          HASHV                         */
1876      /* HASHES USING THE VALUE #'S AND THE    */
1877      /* OPERATOR MOD HASHBASE.                */
1878      /* OP = THE OPERATOR                    */
1879      /* IND = THE INDICATOR FOR THE CURRENT   */
1880      /* EXEC STACK ENTRIES OR VALUE TABLE  */
1881      /* ENTRIES.                            */
1882      /* 0 IS THE CURRENT EXEC STACK         */
1883      /* 1 ALREADY EXISTS IN THE VALUE TABLE */
1884      /* HASH IS FORMED:                     */
1885      /* OPA + ... + OPN + OPERAND + F       */
1886      /* HASHBASE IS CURRENTLY SET TO 127.   */
1887      /*                                     */
1888      /*<><><><><><><><><><><><><><><>*/
1889      HASHV: PROCEDURE (OP, IND) FIXED;
1890      DECLARE (OP, IND, NUMOP, SUM) FIXED;
1891      NUMOP = OP_DEGL(OP);
1892      SUM = OP;
1893      DO WHILE NUMOP > 0;
1894          NUMOP = NUMOP - 1;
1895          DO CASE IND;
1896              SUM = SUM + EXVAL# (ETOP - NUMOP);
1897              SUM = SUM + EXNAM# (ETOP - NUMOP);
1898          END;
1899      END;
1900      RETURN (SUM + IND) MOD HASHBASE;
1901      END HASHV;

```







[illegible]





[illegible]





[illegible]













2431		/* CASE 3: ENI => (TYPE) */
2432	;	
2433		/* CASE 4: XIT => (TYPE) */
2434	;	
2435		/* CASE 5: INT => (TYPE) */
2436		/* CONVERT FROM EXTERNAL TO */
2437		/* INTERNAL FORM */
2438	DO;	
2439	I = EXADD(ETOP);	
2440	CONINT(I) = I_CONVERT(CONSYM(I));	
2441	END;	
2442	;	/* CASE 6: REAL => (TYPE) */
2443		
2444		/* CASE 7: BOOL => (TYPE) */
2445	DO;	
2446	I = EXADD(ETOP);	
2447	IF CONSYM(I) = 'I' THEN CONINT(I) = 1;	
2448	ELSE CONINT(I) = 0;	
2449	END;	
2450		/* CASE 8: IARRAY => (TYPE) */
2451	;	
2452		/* CASE 9: RARRAY => (TYPE) */
2453	;	
2454		/* CASE 10: REFER => (OPCODE) */
2455	;	
2456		/* CASE 11: TOGGLE => (OPCODE) */
2457	;	
2458		/* CASE 12: PASS => (OPCODE) */
2459	;	
2460		/* CASE 13: TRU REAL => */
2461	;	/* INT (OPCODE) */
2462		
2463		/* CASE 14: RND REAL => */
2464	;	/* INT (OPCODE) */
2465		
2466		/* CASE 15: FLT INT => */
2467	;	/* REAL (OPCODE) */
2468		
2469		/* CASE 16: ADD INT INT => */
2470		/* INT (OPCODE) */
2471	DO;	
2472	CALL SET_IJ;	
2473	K = J + I;	
2474	CALL ENTERI;	
2475	END;	
2476		/* CASE 17: RADD REAL REAL => */
2477	;	/* REAL (OPCODE) */
2478		
2479		/* CASE 18: SUB INT INT => */
2480		/* INT (OPCODE) */
2481	DO;	
2482	CALL SET_IJ;	
2483	K = J - I;	
2484	CALL ENTERI;	
2485	END;	
2486		/* CASE 19: RSUB REAL REAL => */
2487	;	/* REAL (OPCODE) */
2488		
2489		/* CASE 20: MUL INT INT => */
2490		/* INT (OPCODE) */
2491	DO;	
2492	CALL SET_IJ;	
2493	K = J * I;	
2494	CALL ENTERI;	
2495	END;	
2496		/* CASE 21: RMUL REAL REAL => */
2497	;	/* REAL (OPCODE) */
2498		
2499		/* CASE 22: DIV INT INT => */
2500		/* INT (OPCODE) */
2501	DO;	
2502	CALL SET_IJ;	
2503	IF I = 0 THEN	
2504	CALL ERROR(' ATTEMPT TO DIVIDE BY ZERO', 3);	
2505	ELSE	
2506	DO;	
2507	K = I / J;	
2508	CALL ENTERI;	
2509	END;	
2510	END;	
2511		/* CASE 23: RDIV REAL REAL => */
2512	;	/* REAL (OPCODE) */
2513		
2514		/* CASE 24: EXP INT INT => */
2515	;	/* INT (OPCODE) */
2516		
2517		/* CASE 25: RIXP REAL INT => */
2518		/* REAL (OPCODE) */





2519	;	
2520		/* CASE 26: IRXP INT REAL => */
2521		/* REAL (OPCODE) */
2522	;	
2523		/* CASE 27: RRXP REAL REAL => */
2524		/* REAL (OPCODE) */
2525	;	
2526		/* CASE 28: LSS DEFAULT */
2527		/* DEFAULT => BOOL */
2528		/* (OPCODEE) */
2529	DO;	
2530	CALL SET_IJ;	
2531	K = J < I;	
2532	CALL ENTERB;	
2533	END;	
2534		/* CASE 29: LEQ DEFAULT */
2535		/* DEFAULT => BOOL (OPCODE) */
2536	DO;	
2537	CALL SET_IJ;	
2538	K = J <= I;	
2539	CALL ENTERB;	
2540	END;	
2541		/* CASE 30: EQI DEFAULT */
2542		/* DEFAULT => BOOL (OPCODE) */
2543	DO;	
2544	CALL SET_IJ;	
2545	K = J = I;	
2546	CALL ENTERB;	
2547	END;	
2548		/* CASE 31: NEQ DEFAULT */
2549		/* DEFAULT => BOOL (OPCODE) */
2550	DO;	
2551	CALL SET_IJ;	
2552	K = J >= I;	
2553	CALL ENTERB;	
2554	END;	
2555		/* CASE 32: GEQ DEFAULT */
2556		/* DEFAULT => BOOL (OPCODE) */
2557	DO;	
2558	CALL SET_IJ;	
2559	K = J > I;	
2560	CALL ENTERB;	
2561	END;	
2562		/* CASE 33: GTR DEFAULT */
2563		/* DEFAULT => BOOL (OPCODE) */
2564	DO;	
2565	CALL SET_IJ;	
2566	K = J > I;	
2567	CALL ENTERB;	
2568	END;	
2569		/* CASE 34: INEG INT => INT */
2570		/* (OPCODE) */
2571	DO;	
2572	I = EXADD(ETOP);	
2573	I = CONINT(I);	
2574	K = -I;	
2575	CALL ENTERB;	
2576	END;	
2577		/* CASE 35: RNEG REAL => REAL */
2578		/* (OPCODE) */
2579	;	
2580		/* CASE 36: NOT BOOL => BOOL */
2581		/* (OPCODE) */
2582	DO;	
2583	I = EXADD(ETOP);	
2584	I = CONINT(I);	
2585	K = ~I;	
2586	CALL ENTERB;	
2587	END;	
2588		/* CASE 37: AND BOOL BOOL > */
2589		/* BOOL (OPCODE) */
2590	DO;	
2591	CALL SET_IJ;	
2592	K = J & I;	
2593	CALL ENTERB;	
2594	END;	
2595		/* CASE 38: BOR BOOL BOOL > */
2596		/* BOOL (OPCODE) */
2597	DO;	
2598	CALL SET_IJ;	
2599	K = J   I;	
2600	CALL ENTERB;	
2601	END;	
2602		/* CASE 39: LOD LOC => */
2603		/* DEFAULT (OPCODE) */
2604	;	
2605		/* CASE 40: STO LOC DEFAULT */
2606		/* => DEFAULT (OPCODE) */



2607	;		/* CASE 41: STD LOC DEFAULT */
2608			/* => (OPCODE) */
2609	;		
2610			/* CASE 42: DEL DEFAULT => */
2611			/* (OPCODE) */
2612	;		
2613			/* CASE 43: DUP => DEFAULT */
2614			/* (OPCODE) */
2615	;		
2616			/* CASE 44: XCH => (OPCODE) */
2617			/* (OPCODE) */
2618	;		
2619			/* CASE 45: HLT => (OPCODE) */
2620			/* (OPCODE) */
2621	;		
2622			/* CASE 46: BRS XIT => */
2623			/* (OPCODE) */
2624	;		
2625			/* CASE 47: BSC BOOL XIT => */
2626			/* (OPCODE) */
2627	DO;		/* CONTROL STACK */
2628			/* */
2629			
2630	I = EXADD(ETOP);		/* IS TOP ENTRY ON EXEC STACK */
2631	I = CONINT(I);		/* A CONSTANT */
2632	IF I THEN		
2633			/* YES...STRIP IT OFF... IE., */
2634	DO;		/* SET UP TO TAKE TRUE BRANCH */
2635			/* */
2636			
2637			/* NO...SET UP TO TAKE THE */
2638			/* FALSE BRANCH */
2639			/* */
2640	SAV TOG(CTOP - 1) = SAV TOG(CTOP);		
2641	CONTROL(CTOP-1)=CONTROL(CTOP);		
2642	END;		
2643			
2644			
2645			
2646	#XITS=#XITS-1;		
2647	CTOP=CTOP-1;		
2648	END;		
2649			/* CASE 48: NOP => (OPCODE) */
2650	;		/* CASE 49: PRO XIT => */
2651			/* (OPCODE) */
2652	;		
2653			/* CASE 50: RTN => (OPCODE) */
2654	;		/* (OPCODE) */
2655			
2656			/* CASE 51: GET INT => LOC */
2657			/* (OPCODE) */
2658	;		
2659			/* CASE 52: RET LOC => */
2660			/* (OPCODE) */
2661	;		
2662			/* CASE 53: RRD => REAL */
2663			/* (OPCODE) */
2664	;		
2665			/* CASE 54: IRO => INT */
2666			/* (OPCODE) */
2667	;		
2668			/* CASE 55: BRD => BOOL */
2669			/* (OPCODE) */
2670	;		
2671			/* CASE 56: WRV DEFAULT => */
2672			/* (OPCODE) */
2673	;		
2674			/* CASE 57: DMP => (OPCODE) */
2675			/* (OPCODE) */
2676	;		
2677			/* CASE 58: TAB INT => */
2678			/* (OPCODE) */
2679	;		
2680			/* CASE 59: SUP DEFAULT => */
2681			/* (OPCODE) */
2682	END;		
2683	RETURN RESULT;		
2684	END PROPAGATE; /*		
2685			



[illegible]















```

2959 DO WHILE (NUMOP < DEGL) & MATCH;
2960 EXPNT = ETOP - NUMOP;
2961 RLPNT = RLPNT + 1;
2962 DATA = OP_INFO(RLPNT);
2963 IND = DATA & "1";
2964 ID# = SHR(DATA, 1);
2965 IF IND THEN
2966     /* IS OPERAND RULE FOR AN
2967     /* IDENTITY CONSTANT
2968     /*
2969     DO;
2970     MATCH = FALSE;
2971     IND = OP_INFO(OPFORM);
2972     /* YES...
2973     /* IS EXEC STACK OPERAND A
2974     /* CONSTANT AND OF PROPER
2975     /* TYPE FOR EXPRESSION.
2976     /*
2977     IF EXCON(EXPNT) & (EXTYPE(EXPNT) = IND) THEN
2978     DO;
2979     /* YES...ADDRESS OF THE
2980     /* IDENTITY IN CONSTANT TABLE
2981     /*
2982     CONSADR = LOOKCONS(OP_STR(ID#), IND);
2983     /*
2984     /* MATCH IF EXEC CONSTANT IS
2985     /* EQUAL TO THE IDENTITY IN
2986     /* THE RULE.
2987     /*
2988     MATCH = (CONSADR ^= 0) & (CONSADR = EXADD(EXPNT));
2989     END;
2990     /*
2991     /* NOT A CONSTANT, IE...
2992     /* MUST BE VARIABLE OPERAND.
2993     /*
2994     END; ELSE
2995     /* ALWAYS TRUE AT LEAST ONCE
2996     /* ASKING HAS EXEC STACK BEEN
2997     /* SAVED.
2998     /*
2999     IF POSIT < ID# THEN
3000     DO;
3001     /* NO...SAVE EXEC STACK
3002     /* ELEMENT AT TOP OF EXEC STK
3003     /*
3004     POSIT = ID#;
3005     CALL GET_EXECP;
3006     EXCON(ETOPR) = EXCON(EXPNT);
3007     EXADD(ETOPR) = EXADD(EXPNT);
3008     EXVAL#(ETOPR) = EXVAL#(EXPNT);
3009     STATES(STLOC + ID#) = ETOPR;
3010     END; ELSE
3011     DO;
3012     MATCH = FALSE;
3013     STPNT = STATES(STLOC + ID#);
3014     /*
3015     /* IS THE SAVED VARIABLE EQUAL
3016     /* TO THE RESULT VARIABLE
3017     /*
3018     IF EXVAL#(STPNT) = EXVAL#(EXPNT) THEN
3019     MATCH = TRUE;
3020     ELSE
3021     /*
3022     /* ARE EITHER CURRENT OR
3023     /* SAVED OPERANDS CONSTANTS
3024     /*
3025     IF EXCON(EXPNT) | EXCON(STPNT) THEN
3026     DO;
3027     /*
3028     /* YES...THE SAME CONSTANT
3029     /*
3030     IF EXADD(STPNT) ^= EXADD(EXPNT) THEN
3031     DO;
3032     /*
3033     /* NO...CLEAR THE CONSTANT
3034     /* ENTRY IN THE SAVED ENTRY
3035     /* IE..VARIABLE OPERAND
3036     /*
3037     EXCON(STPNT) = 0;
3038     EXADD(STPNT) = 0;
3039     END;
3040     END;
3041     END;
3042     /*
3043     /* ADVANCE COUNTER AND FORMAL
3044     /* TYPE TO SEARCH FOR THEN
3045     /* BACK FOR NEXT OPERAND
3046     /*
3047     NUMOP = NUMOP + 1;

```



```

3047      UPFORM = UPFORM + 1;
3048
3049      /* END OF (NUMOP < DEGL) &
3050      /* MATCH.
3051      /*
3052  END;
3053  CONSADR = RLPNT;
3054
3055      /* WAS A MATCH FOUND
3056      /*
3057  IF MATCH THEN
3058
3059      /* YES...CLEAR SIMPLIFICATION
3060      /* POINTER.
3061      /*
3062  RLPNT = 0; ELSE
3063  DO;
3064
3065      /* NO...IS OPERATOR
3066      /* COMMUNITIVE
3067      /*
3068  IF COMOP THEN
3069  DO;
3070
3071      /*
3072      /*
3073      /* YES...LETS TRY A
3074      /* PERMUTATION.
3075      OPADR = ADDR(OP_INFO(SAVRLPNT + 1));
3076      FIRSTPERM = PERMUTE(DEGL, OPADR, 2, TRUE);
3077      IF FIRSTPERM THEN
3078      RLPNT = SAVRLPNT;
3079      ELSE
3080      DO;
3081      FIRSTCALL = TRUE;
3082      RLPNT = TYP;
3083  END;
3084  END;
3085
3086      /* OPERATOR IS NOT COMMUNITIV
3087      /*
3088  ELSE
3089  DO;
3090      RLPNT = TYP;
3091      FIRSTCALL = TRUE;
3092  END;
3093
3094      /* IF NEW RULE SELECTED,
3095      /* RESET NEXT RULE POINTER.
3096      /*
3097  IF TYP = RLPNT THEN TYP = OP_INFO(RLPNT);
3098  END;
3099
3100      /* BACK TO TEST FOR NEXT RULE
3101      /* OR TEST A PERMUTATION FOR
3102      /* A COMMUNITIVE OPERATOR
3103      /* AGAINST SAME RULE.
3104      /*
3105  END;
3106  RLPNT = CONSADR;
3107  IF MATCH THEN
3108      /* WAS A SIMPLIFICATION RULE
3109      /* MATCHED.
3110
3111  DO;
3112      OPTIM_TYPE(M_CONTROL) = "1";
3113
3114      /* YES...CHECK THE RESULTANT
3115      /* TYPE, ETC.
3116      /*
3117  RLPNT = RLPNT + 1;
3118  RESTYP = OP_INFO(RLPNT);
3119  RESPNT = RESTYPE;
3120  IND = RESTYP & "1";
3121  RESTYP = SHR(RESTYP, 1);
3122
3123      /* SHOULD THE RESULT BE AN
3124      /* IDENTITY
3125      /*
3126  IF IND THEN
3127  DO;
3128
3129      /* YES...IS THE IDENTITY IN
3130      /* THE CONSTANT TABLES
3131      /*
3132  CONSADR = LOOKCONS(OP_STR(RESTYP), RESPNT);
3133  IF CONSADR = 0 THEN
3134  DO;
3135
3136      /* NO...PUT IDENTITY IN
3137      /* CONSTANT TABLE.
3138      /*
3139  CALL ENTERCONS(OP_STR(RESTYP), RESPNT);
3140  CONSADR = CONLOC;

```











```

3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310

```

```

DO;
  N = 0;
  IF TYP <= XIT THEN
    DO CASE TYP;
      CALL CERROR('TYPE NULL ENCOUNTERED');
      CALL CERROR('TYPE DEFAULT ENCOUNTERED');
      /*
      /*      0 = NULL
      /*      1 = DEFAULT
      /*      2 = LOC
      /*      3 = ENT
      /*      4 = XIT
      /*      5 = INT
      /*      6 = REAL
      /*      7 = BOOL
      /*      8 = IARRAY
      /*      9 = RARRAY
      /*
      /* SPECIAL CASES
      /* NULL,DEFAULT,LOC,ENT,XIT
      /*
      /*      LOC
      /* LOC TYPE CONSTANT
      /*
      /* IS THIS A CONSTANT
      /*
      IF C THEN
        /*
        /* YES...
        /*
        V = 0;
        ELSE
        DO;
          /* NO...IDENTIFIER SHOULD BE
          /* IN ADR TABLE
          /*
          IF A = 0 THEN
            CALL CERROR('NULL LOCATION ');
            /*
            /* GO FIND IT
            /*
            I = LOOKADD(A);
            IF I = 0 THEN
              /* IDENTIFIER NOT PREVIOUSLY
              /* DEFINED
              /*
              DO;
                /*
                /* GET NEW ADDRESS FOR THIS
                /* IDENTIFIER
                /*
                CALL ENTERADD(A);
                I = ADDLOC;
                ADDTYPE(I) = NULL;
                ADDVAL(I) = 0;
                ADDCON(I) = 0;
                V, ADDNAM(I) = GETVAL#(FORMAL);
              END;
            ELSE V = ADDNAM(I);
          END;
          /*
          /*      ENT
          /* ENT TYPE CONSTANT.. ENTER
          /* SET NEW BLOCK NUMBER ON
          /* THE CONTROL TABLE FOR
          /* FUTURE PROCESSING AND EXIT
          /* BACK TO CONTROL FLOW.
          /*
          DO;
            CALL INSRT_SUC_BLOCK(M_CONTROL);
            RETURN;
          END;
          /*
          /*      XIT
          /* XIT TYPE CONSTANT..SET NEW
          /* BLOCK NUMBER ON CONTROL
          /* TABLE FOR FUTURE PROCESSIN
          /* WITH ENTRY ADDRESS A
          /*
          CALL INSRT_SUC_BLOCK(A);
          END; ELSE
          /*
          /* REFERENCE IS TO A STANDARD
          /* DECLARATION - BEGIN
          /* EXAMINATION FOR EXEC STACK
          /* SET UP.
          /*
          IF A = 0 THEN
            /* REFERENCE TO INDETERMINATE
            /* TYPE - I.E., NO IDENTIFIER
            /* PRESENT
            /*
            DO;
              C = 0;
              V = GETVAL#(COMPU);
              N = GETVAL#(FORMAL);
            END; ELSE

```



```

311
312
313
314
315 DO;
316     C = 1;
317     V = CONVAL(A);
318     N = 0;
319
320 END;
321 IF TYP = XIT THEN
322
323     DO;
324         CALL GET_EXEC;
325         I = ETOP;
326         EXNAM#(I) = N;
327         EXTYPE(I) = TYP;
328         EXCON(I) = C;
329         EXADO(I) = A;
330         EXVAL#(I) = V;
331     END;
332
333 ELSE
334
335     DO;
336         DEGL = OP_DEGL(OP);
337         DEGR = OP_DEGR(OP);
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```



253





```

3481                                     /* TO THE IDENTIFIER */
3482                                     /* */
3483 V = EXVAL#(ETOP - NUMOP);
3484                                     /* */
3485                                     /* COMPARE THE NEW VALUE# */
3486                                     /* AGAINST THE OLD EXPRESSION */
3487                                     /* */
3488 N = A - J;
3489 IF VTAB(N) = V THEN
3490 COUNT = COUNT + 1;
3491 ELSE
3492                                     /* */
3493                                     /* SET IN NEW VALUE # FOR */
3494                                     /* OPERAND IF CHANGED */
3495                                     /* */
3496 VTAB(N) = V;
3497 J = J + 1;
3498 END;
3499                                     /* */
3500                                     /* IF AN OPERAND HAS CHANGED */
3501                                     /* AND THE OPTIM WAS A COM */
3502                                     /* SUB ELLIM THEN RESET IT TO */
3503                                     /* BE NOT OPTIMIZED. */
3504                                     /* */
3505 J = COUNT = DEGL;
3506 SUBEXP = (OPTIM_TYPE(M_CONTROL) = "3") |
3507          (OPTIM_TYPE(M_CONTROL) = "4");
3508 IF (~ J) & SUBEXP THEN
3509 OPTIM_TYPE(M_CONTROL) = "0";
3510 I, V = 0;
3511 N = A & "7FFF";
3512                                     /* */
3513                                     /* SEE IF AN IDENTICAL */
3514                                     /* EXPRESSION EXISTS IN THE */
3515                                     /* VALUE TABLE */
3516                                     /* */
3517 I = LOOKUPV(OP, COMPU, VTOP);
3518                                     /* */
3519 IF I > 0 THEN
3520                                     /* YES...SET IN THE NEW */
3521 DO;
3522                                     /* EXPRESSION RESULT INTO */
3523                                     /* THE OLD EXPRESSION ENTRY */
3524                                     /* AND MARK AS COMM SUB ELLIM */
3525                                     /* */
3526 OPTIM_TYPE(M_CONTROL) = "3";
3527 A = A | SHL(I, 15);
3528 VPOINT = I + DEGL + 2;
3529 V, VTAB(N - DEGL - 1) = VTAB(VPOINT);
3530 END; ELSE
3531 DO;
3532                                     /* */
3533                                     /* NO..HAS THE EXP ALREADY */
3534                                     /* BEEN PROCESSED AND NO */
3535                                     /* OPERANDS HAVE CHANGED */
3536                                     /* */
3537 IF C & J THEN
3538                                     /* */
3539                                     /* GET THE OLD RESULT VAL# */
3540                                     /* */
3541 V = VTAB(N - DEGL - 1);
3542 ELSE
3543                                     /* */
3544                                     /* NO..GET A NEW VALUE # */
3545                                     /* AND MARK AS PROCESSED */
3546                                     /* */
3547 CALL CONSTANT_CLASS;
3548                                     /* */
3549                                     /* CHAIN NEW VALUE TABLE */
3550                                     /* ENTRY TO SAME CLASS AND */
3551                                     /* SET IN RESULT VALUE # */
3552                                     /* */
3553 I = REENTERV(OP, COMPU);
3554 VPOINT = I + DEGL + 2;
3555 VTAB(VPOINT) = V;
3556 END;
3557 I = V;
3558 V = 0;
3559                                     /* */
3560                                     /* IS PROPAGATION REQUESTED */
3561                                     /* */
3562 IF PROPTOG THEN
3563                                     /* */
3564                                     /* YES...TRY TO PERFORM A */
3565                                     /* SIMPLIFICATION */
3566                                     /* */
3567 CALL SIMPLIFY;
3568                                     /* */
3569                                     /* IF THE OPTIM IS COM SUB ... */
3570                                     /* */

```





```

3575                                     /* EXP AND NO SIMPLIFICATION */
3576                                     /* WAS DETECTED...BYPASS */
3577                                     /* CLEARING. */
3578                                     /* */
3579                                     /* */
3580                                     /* IF NEW VALUE # ASSIGNED, */
3581                                     /* SET IT INTO VTAB */
3582                                     /* */
3583                                     /* */
3584 IF (V = 0) & (OPTIM_TYPE(M_CONTROL) = "3") THEN
3585 V = 1;
3586 ELSE
3587 DO;
3588 IF V = 0 THEN
3589 DO;
3590 OPTIM_TYPE(M_CONTROL) = "0";
3591 V = 1;
3592 END;
3593 END;
3594 VTAB(N - DEGL - 1) = V;
3595 VTAB(VPOINT) = V;
3596 SUBEXP = (OPTIM_TYPE(M_CONTROL) = "2") |
3597           (OPTIM_TYPE(M_CONTROL) = "4");
3598 IF SUBEXP THEN
3599 ADDCON(VPOINT) = EA;
3600 I, N = 0;
3601 END; ELSE
3602                                     /* NO ADDRESS AVAILABLE. */
3603                                     /* CHECK FOR FIRST PASS THIS */
3604                                     /* BLOCK */
3605                                     /* */
3606 IF PASS# = 1 THEN
3607 DO;
3608                                     /* YES...CHECK TO SEE IF AN */
3609                                     /* OPTIMIZATION IS POSSIBLE. */
3610                                     /* BUILD A VTAB ENTRY FOR */
3611                                     /* THIS EXPRESSION */
3612                                     /* */
3613 NUMOP = DEGL;
3614 COUNT = 0;
3615 A = VTOPR - 1;
3616                                     /* SET TO TEST ALL OPERANDS */
3617                                     /* */
3618 DO WHILE NUMOP > 0;
3619 N = DEGL - NUMOP;
3620 NUMOP = NUMOP - 1;
3621 NT = EXVAL#(ETOP - N);
3622                                     /* CHECK IF VALUE # PRESENT */
3623                                     /* FOR THE OPERANDS ON THE */
3624                                     /* EXEC STACK. IE., A */
3625                                     /* USEFUL DATA IN OPERAND */
3626                                     /* CHECK. */
3627                                     /* */
3628 IF NT != 0 THEN
3629 DO;
3630                                     /* YES...SET INTO TOP OF */
3631                                     /* VALUE TABLE. IE., SAVED */
3632                                     /* OPTIMIZATION INFORMATION */
3633                                     /* AREA. */
3634                                     /* */
3635 COUNT = COUNT + 1;
3636 CALL GET VTOPR;
3637 VTAB(VTOPR) = NT;
3638 END;
3639 END;
3640 IF COUNT != DEGL THEN
3641 DO;
3642 VTOPR = A + 1 + DEGL;
3643 A = 0;
3644 I = 0;
3645 END; ELSE
3646                                     /* DID ALL OPERANDS HAVE */
3647                                     /* A VALUE #...INDICATES ALL */
3648                                     /* OPERANDS ARE USEFUL - NOW */
3649                                     /* CHECK FOR A COMMON */
3650                                     /* SUBEXPRESSION ELIMINATION. */
3651                                     /* */
3652 DO;
3653                                     /* YES...SEE IF A VALUE TABLE */
3654                                     /* ENTRY EXISTS THAT IS */
3655                                     /* IDENTICAL */
3656                                     /* */

```



```

CALL GET_VTOPR;
I = LOOKUPV(OP, COMPU, VTOP);
/* YES...IDENTICAL ENTRY */
IF I > 0 THEN
DO;
/* EXISTS - GET THE VALUE #
/* OF ITS RESULT AND MARK
/* THIS INSTRUCTION AS A
/* COMMON SUBEXPRESSION
/* ELIMINATION.
/*
VPOINT = I + DEGL + 2;
A = A | SHL(1, 15);
V = VTAB(VPOINT);
OPTIM_TYPE(M_CONTROL) = "3";
END; ELSE
DO;
/* NO..GET A NEW VALUE# , SET
/* OPTIM BIT AND BUILD A NEW
/* EXPRESSION ENTRY.
/*
RESIND = COMPU;
HASHADR = HASHV(OPERTR, COMPU);
EADR = ETOP;
V = GETVAL#(COMPU);
A = A | SHL(1, 15);
I = ENTERV;
VPOINT = I + DEGL + 2;
VTAB(VPOINT) = V;
END;
/* SET LOCATION INTO TOP OF
/* VTAB.
/*
VTAB(VTOPR) = M_CONTROL;
I = V;
CALL GET_VTOPR;
V = 0;
/* CHECK FOR PROPAGATION
/* REQUESTED
/*
IF PROPTOG THEN
/* YES...TRY TO PERFORM A
/* SIMPLIFICATION
/*
CALL SIMPLIFY;
/* SAVE THE RESULT VALUE#
/*
IF (V = 0) & (OPTIM_TYPE(M_CONTROL) = "3") THEN
V = I;
ELSE
DO;
IF V = 0 THEN
DO;
OPTIM_TYPE(M_CONTROL) = "0";
V = I;
END;
END;
VTAB(VPOINT) = V;
VTAB(VTOPR) = V;
SUBEXP = (OPTIM_TYPE(M_CONTROL) = "2") |
(OPTIM_TYPE(M_CONTROL) = "4");
IF SUBEXP THEN
ADDCON(VPOINT) = EA;
END; ELSE
DO;
/* PASS# IS NOT EQUAL TO 1 */
A = 0;
I = 0;
V = 0;
END;
N = 0;
/* SET IN THE ADDRESS OF VTAB*
/* CONTAINING OPA
/* OPB
/* OPTIM ADR
/* RESULT
/* SET A INTO CODE FILE. THIS
/* IS THE POINT OF MARKING
/* OPTIMIZATION FOR LATER
/* DETECTION.
/*
CSFCODE(M_CONTROL) = (CODE & "FFFF8000") | A;
N = A & "7FFF";

```



```

3751 IF OPTIM_TYPE(M_CONTROL) = "1" THEN
3752 CALL SET_OPTIM_LOC(N);
3753 ELSE
3754 DO;
3755     IF N = 0 THEN
3756     DO;
3757         N = N - DEGL;
3758         VTAB(N) = M_CONTROL;
3759     END;
3760 END;
3761 N = 0;
3762
3763 /* NOW REDUCE THE EXEC STACK */
3764 /* ACCORDING TO THE # OF */
3765 /* OPERANDS */
3766 /*
3767
3768 ETOP = ETOP - DEGL;
3769 IF DEGR > 0 THEN /* BUILD AN EXEC STACK ENTRY */
3770 /* FOR THE RESULT */
3771 DO;
3772     I, ETOP = ETOP + 1;
3773     EXCON(I) = EA = 0;
3774     EXTYPE(I) = RESTYPE;
3775     EXADD(I) = EA;
3776     EXVAL(I) = V;
3777     EXNAM(I) = 0;
3778 END;
3779 END;
3780 TYPE1_OP: /* END OF SIMPLE OPERATOR */
3781 /* TYPE 1 OPERATOR NOT USED */
3782 /*
3783 CALL CERROR('UNDEFINED OPERATOR TYPE - TYPE 1');
3784 /*
3785 /* CASE 2 LOAD OPERATOR
3786 /* MUST BE LOCATION TYPE
3787 /* VARIABLE ON THE TOP OF THE
3788 /* EXEC STACK TO ALLOW A LOAD
3789 /* FROM THE ADDRESS ON TOP
3790 /* OF STACK.
3791 /*
3792 LOD_OP:
3793 DO;
3794     IF EXTYPE(ETOP) = LOC THEN
3795     DO;
3796         CALL CERROR('ATTEMPT TO LOAD FROM NON-LOC TYPE');
3797         RETURN;
3798     END;
3799     A = EXADD(ETOP);
3800     IF EXCON(ETOP) THEN /* CHECK FOR VALUE IN THE */
3801         /* CONSTANT TABLE */
3802     DO;
3803         /*
3804         /* YES...SET VARIABLES TO
3805         /* INDICATE CONSTANT TO BE
3806         /* LOADED
3807         /*
3808         V = CONVAL(A);
3809         C = TRUE;
3810         TYP = CONTYPE(A);
3811     END; ELSE
3812     DO;
3813         /* NO...FIND IDENTIFIER IN
3814         /* ADDRESS TABLE
3815         /*
3816         I = LOOKADD(A);
3817         V = ADDVAL(I);
3818         IF C THEN /* CHECK FOR PREVIOUSLY
3819             /* ASSIGNED CLASS NUMBER.
3820             /* YES...MARK LOCATION
3821             /*
3822         V, ADDVAL(I) = GETVAL#(COMPU);
3823         ELSE
3824         IF V = 0 THEN /* NO...CHECK FOR VALUE #
3825             /* ASSIGNED
3826         DO;
3827             /*
3828             /* NO...ASSIGN BY CODE
3829             /* LOCATION AND FLAG THE CODE
3830             /* FILE
3831             /*
3832             V, ADDVAL(I) = GETVAL#(COMPU);
3833             CSFCODE(M_CONTROL) = (CODE & "FFFF0000")|SHL(1, 15);
3834             /*
3835             /* MAY WANT TO PUT TYPE IN
3836             /* THIS FIELD LATER.
3837             /*
3838         END;

```



```

3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926

/* MARK TYP AND A FROM ADDRESS */
/* TABLE AND SET C BY VALUE */
/* TABLE PRESENT OR NOT */
/*
TYP = ADDTYPE(I);
A = ADDCON(I);
C = A - 0;
END;

/* SET TOP OF EXEC STACK TO */
/* LOAD EITHER CONSTANT OR */
/* VALUE AS PREVIOUSLY */
/* DETERMINED */
/*
EXNAM#(ETOP) = EXVAL#(ETOP);
EXCON(ETOP) = C;
EXTYPE(ETOP) = TYP;
EXADD(ETOP) = A;
EXVAL#(ETOP) = V;
END;

/* CASE 3 STORE OPERATOR */
/* CAUSES SECOND OPERAND */
/* ON STACK TO BE STORED IN */
/* LOCATION SPECIFIED BY */
/* ADDRESS ON TOP OF STACK. */
/*
STORE_OP:
DO;

/* TOP OF EXEC STACK MUST BE */
/* LOC */
/*
IF EXTYPE(ETOP - 1) - LOC THEN
DO;
CALL CERROR('ATTEMPT TO STORE INTO NON-LOC VARIABLE');
RETURN;
END;
A = EXADD(ETOP - 1);

/* IF NO ADDRESS, ASSUME AN */
/* ARRAY SUBSCRIPT */
/*
IF A = 0 THEN
DO;
CALL CERROR('ARRAY STORES NOT IMPLEMENTED IN CSF');
RETURN;
END;

/* FIND ADDR TABLE LOCATION */
/*
I = LOOKADD(A);

/* NOT IN ADDRESS TABLE */
/*
IF I = 0 THEN
DO;
CALL CERROR('ADDRESS TABLE ERROR (2)', 2);
RETURN;
END;
IF I <= TADDLOC THEN
/* ADDRESS SET IN ANOTHER */
/* BBLOCK */
DO;

/* BUILD A NEW ADDRESS TABLE */
/* ENTRY */
/*
N = ADDNAM(I);
CALL ENTERADD(A);
I = ADDLOC;
ADDNAM(I) = N;
END;
N = ETOP;
ALTAD = 1;
IF EXCON(N) THEN
/* CONSTANT ASSIGNMENT */
/* SET ADDRESS TABLE FROM */
/* CONSTANT TABLE */
DO;
J = EXADD(N);
ADDCON(I) = J;
ADDVAL(I) = EXVAL#(N);
ADDTYPE(I) = CONTYPE(J);
ADDNAM(I) = 0;
END; ELSE
/* NOT A CONSTANT */
/* SET ADDRESS TABLE FROM */
/* EXEC STACK */
ADDCON(I) = 0;.....

```













```

107 SET_UP_BLOCK: PROCEDURE;
108 DECLARE (I,J,K,L,M) FIXED;
109 DECLARE (O,V) FIXED;
110
111                                     /*
112                                     /* FORM POINTER TO STATES
113                                     /* TABLE CURRENT BLOCK
114                                     /*
115
116 ALTEX = 1;
117 ETOP = 0;
118
119                                     /*
120                                     /* RESTORE THE ADDRESS TABLE
121                                     /* FROM THE STATE VECTOR
122                                     /*
123
124 ALTVL = 1;
125 ALTAD = 1;
126 ADDLOC = 0;
127
128                                     /*
129                                     /* REMOVE ALL VTAB ENTRIES
130                                     /* ABOVE HASHBASE
131                                     /*
132
133 CALL REMOVEV(HASHBASE);
134 I = CSFCODE(CURBLOCK) & "FFFF";
135 K = I + 2;
136 J = STATES(K);
137 K = K + 1;
138 I = K + J;
139
140                                     /*
141                                     /* REBUILD THE CURRENT
142                                     /* OPTIMIZE STATE FOR THE
143                                     /* BLOCK.
144                                     /*
145
146 DO WHILE I > K;
147     I = I - 1;
148     J = STATES(I) & "7FFF";
149     L = ADDRESS(J);
150     IF SHR(L, 15) THEN
151
152                                     /* EXEC STACK ENTRY
153                                     /* ADDRESS = 111 ETOP
154                                     /* PUT THE EXEC STACK ENTRY
155                                     /* AT THE TOP OF THE EXEC
156                                     /* STACK AREA.
157
158     DO;
159         L = L & "7FFF";
160         IF ETOP <= L THEN
161             ETOP = L + 1;
162         L = MAXEXEC - L;
163         EXTYPE(L) = ADDTYPE(J);
164         EXVAL#(L) = ADDVAL(J);
165         M = ADDCON(J);
166         EXCON(L) = SHR(M, 15);
167         EXADD(L) = M & "7FFF";
168         EXNAM#(L) = ADDNAM(J);
169     END; ELSE
170     IF L = 0 THEN
171
172                                     /* VALUE TABLE ENTRY
173                                     /* ADDRESS = 0
174                                     /* PUT THE ADDRESS OF THE
175                                     /* VTAB ENTRY ON THE EXEC
176                                     /* STACK.
177
178     DO;
179         J = ADDTYPE(J);
180         L = ETOP;
181         V = ADDVAL(J);
182         J = ADDNAM(J);
183         M = OP_DEGL(O);
184         DO WHILE M > 0;
185             M = M - 1;
186             ETOP = ETOP + 1;
187             EXVAL#(ETOP) = VTAB(J);
188             J = J - 1;
189         END;
190
191                                     /* BUILD A VALUE TABLE ENTRY
192                                     /* FOR EACH TYPE OF
193                                     /* EXPRESSION.
194
195         M = REENTERV(O, COMPU);
196         ETOP = L;
197         M = M + 2 + OP_DEGL(O);
198         VTAB(M) = V;
199     END; ELSE
200
201                                     /*
202                                     /* ADDRESS TABLE ENTRY ;
203                                     /* SIMPLY RESTORES THE ADDRES
204                                     /* TABLE FOR THIS BLOCK
205                                     /*
206
207     DO;
208         CALL ENTERADD(L);

```





```

4191      ADDTYPE(ADDLOC) = ADDTYPE(J);
4192      ADDVAL(ADDLOC) = ADDVAL(J);
4193      ADDCON(ADDLOC) = ADDCON(J);
4194      ADDNAM(ADDLOC) = ADDNAM(J);
4195      END;
4196  END;
4197
4198      /* NOW FILL THE EXEC STACK
4199      /* MOVES THE EXEC DATA FROM
4200      /* THE TOP OF EXEC AREA INTO
4201      /* WORKING EXEC STACK.
4202      */
4203
4204  L = ETOP;
4205  M = MAXEXEC;
4206  DO WHILE L > 0;
4207      EXTYPE(L) = EXTYPE(M);
4208      EXVAL#(L) = EXVAL#(M);
4209      EXCON(L) = EXCON(M);
4210      EXADD(L) = EXADD(M);
4211      EXNAM#(L) = EXNAM#(M);
4212      M = M - 1;
4213      L = L - 1;
4214  END;
4215  END SET_UP_BLOCK;
4216
4217      /*
4218      /* SWAP_BLOCKS
4219      /* THIS ROUTINE MOVES THE ENTRIES IN THE
4220      /* CONTROL LIST. THE ENTRIES MOVED ARE AT
4221      /* ADDRESS A AND B IN THE CONTROL STACK.
4222      */
4223  SWAP_BLOCKS: PROCEDURE (A, B);
4224      DECLARE (A, B, TEMFIX) FIXED, TEM8 BIT(8);
4225      TEMFIX = CONTROL(B);
4226      CONTROL(B) = CONTROL(A);
4227      CONTROL(A) = TEMFIX;
4228      TEM8 = SAV_TOG(B);
4229      SAV_TOG(B) = SAV_TOG(A);
4230      SAV_TOG(A) = TEM8;
4231  END SWAP_BLOCKS;
4232
4233      /*
4234      /* ORDER_BLOCK_LIST
4235      /* THIS ROUTINE SETS UP THE CONTROL STACK
4236      /* BY DIFFERENT ORDERING ALGORITHMS
4237      /* DEPENDENT UPON SELECT_FLAG.
4238      /* 0 = CANNONICAL (LIFO)
4239      /* 1 = STEEPEST DESCENT
4240      /* (HEURISTICALLY THIS IS DONE
4241      /* BY TAKING THE BLOCK WITH THE
4242      /* LEAST # OF STATES AND MOVING
4243      /* IT TO BE PROCESSED NEXT).
4244      /* 2 = FIFO
4245      /* 3 = SELECTS THE BLOCK WITH THE
4246      /* MAXIMUM CURRENT STATE.
4247      /* ROUTINES CALLED:
4248      /* SWAP_BLOCKS
4249      */
4250  ORDER_BLOCK_LIST: PROCEDURE;
4251      DECLARE (CODEADR, BLKHDADRA, BLKHDADRB, STATEA, STATEB, NUMSTATESA,
4252              NUMSTATSB, I, L, J, K, FOUND, DATA) FIXED;
4253      DO CASE SELECT_FLAG;
4254      CANNONICAL:
4255
4256      /*
4257      /* CANNONICAL REQUESTED
4258      /* CONTROL STACK IS ALREADY
4259      /* BUILT PROPERLY FOR LIFO
4260      /*
4261      RETURN;
4262
4263      /*
4264      /* STEEPEST DESCENT REQUESTED
4265      /*
4266  STEEPEST_DESCENT:
4267      DO;
4268      /* ONLY ONE ENTRY ON STACK
4269      /* YES...EXIT STACK MUST BE
4270      /* RIGHT
4271      /*
4272      /* NO...SORT STACK INTO
4273      /* DESCENDING ORDER
4274      /*
4275      J = 1;
4276      DO WHILE J < CTOP;
4277          CODEADR = CONTROL(J);
4278

```





```

4279 CODEADR = CODEADR & "7FFF";
4280 BLKHDADRA = CSFCODE(CODEADR);
4281 BLKHDADRA = BLKHDADRA & "7FFF";
4282 CODEADR = CONTROL(J + 1);
4283 CODEADR = CODEADR & "7FFF";
4284 BLKHDADRB = CSFCODE(CODEADR);
4285 BLKHDADRB = BLKHDADRB & "7FFF";
4286 STATEA = CSFCODE(BLKHDADRA);
4287 STATEA = STATEA & "7FFF";
4288 STATEB = CSFCODE(BLKHDADRB);
4289 STATEB = STATEB & "7FFF";
4290 STATEA = STATEA + 2;
4291 STATEB = STATEB + 2;
4292 NUMSTATESA = STATES(STATEA);
4293 NUMSTATESB = STATES(STATEB);
4294
4295 /* IN DESCENDING ORDER */
4296 /*
4297 IF NUMSTATESA < NUMSTATESB THEN
4298 DO;
4299 CALL SWAP_BLOCKS(J, J + 1);
4300 K = 1;
4301 L = J;
4302
4303 /* BUBBLE B TO PROPER PLACE */
4304 /*
4305 I = J - 1;
4306 DO WHILE I > 0;
4307 CODEADR = CONTROL(I);
4308 CODEADR = CODEADR & "7FFF";
4309 BLKHDADRA = CSFCODE(CODEADR);
4310 BLKHDADRA = BLKHDADRA & "7FFF";
4311 STATEA = CSFCODE(BLKHDADRA);
4312 STATEA = STATEA & "7FFF";
4313 STATEA = STATEA + 2;
4314 NUMSTATESA = STATES(STATEA);
4315 IF NUMSTATESA < NUMSTATESB THEN
4316 DO;
4317 CALL SWAP_BLOCKS(I, L);
4318 L = I;
4319 END;
4320 I = I - 1;
4321 END;
4322 END;
4323 J = J + 1;
4324 END;
4325 FIRST_IN_FIRST_OUT:
4326 DO;
4327
4328 /* IF ONLY ONE ENTRY, LIST IS */
4329 /* ALREADY GOOD */
4330 /*
4331 IF CTOP <= 1 THEN
4332 RETURN;
4333 FOUND = FALSE;
4334 J = 1;
4335 DO WHILE ~ FOUND;
4336 DATA = CONTROL(J);
4337
4338 /* IS BLOCK GOOD */
4339 /*
4340 IF ~(SHR(DATA, 31) & "1") THEN
4341 DO;
4342 FOUND = TRUE;
4343 K = TRUE;
4344 DO WHILE K;
4345 CALL SWAP_BLOCKS(J, J + 1);
4346 IF J = (CTOP - 1) THEN
4347 K = FALSE;
4348 J = J + 1;
4349 END;
4350 END;
4351 J = J + 1;
4352 IF J >= CTOP THEN
4353 RETURN;
4354 END;
4355 END;
4356 MAX_POOL:
4357 DO;
4358 IF CTOP <= 1 THEN
4359 RETURN;
4360
4361 /* ONLY ONE ENTRY ON STACK */
4362 /* YES...EXIT STACK MUST BE */
4363 /* RIGHT */
4364 /*
4365 /* NO...SORT STACK INTO */
4366 /* ASCENDING ORDER */
4367 /*
J = 1;
DO WHILE J < CTOP;

```



```
/* GET THE STATE POINTER */
```

```

CODEADR = CONTROL(J);
CODEADR = CODEADR & "7FFF";
BLKHDADR = CSFCODE(CODEADR);
BLKHDADR = BLKHDADR & "7FFF";
CGDEADR = CONTROL(J + 1);
CODEADR = CODEADR & "7FFF";
BLKHDADRB = CSFCODE(CODEADR);
BLKHDADRB = BLKHDADRB & "7FFF";
STATEA = CSFCODE(BLKHDADR);
STATEA = STATEA & "7FFF";
STATEB = CSFCODE(BLKHDADRB);
STATEB = STATEB & "7FFF";
STATEA = STATEA + 2;
STATEB = STATEB + 2;
NUMSTATESA = STATES(STATEA);
NUMSTATESB = STATES(STATEB);
IF NUMSTATESA > NUMSTATESB THEN
DO;

```

```
CALL SWAP_BLOCKS(J, J + 1);
K = 1;
L = J;
```

```

/*                                     */
/* BUBBLE 8 TO PROPER PLACE          */
/*                                     */

```

```

I = J - 1;
DO WHILE I > 0;
    CODEADR = CONTROL(I);
    CODEADR = CODEADR & "7FFF";
    BLKHOADRA = CSFCODE(CODEADR);
    BLKHOADRA = BLKHOADRA & "7FFF";
    STATEA = CSFCODE(BLKHOADRA);
    STATEA = STATEA & "7FFF";
    STATEA = STATEA + 2;
    NUMSTATESA = STATES(STATEA);
    IF NUMSTATESA > NUMSTATESB THEN
        DO;
            CALL SWAP_BLOCKS(I, L);
            L = I;
        END;
    END;
END;

```

```
END;  
I = I - 1;
```

```
/* END WHILE K                                */
/* END FIRST A < B                             */
```

```

END;
J = J + 1;

```

```
/* DO WHILE          */
/* MAX POOL          */
/* OF ALL CASES      */
```

```

RETURN,
END ORDER_BLOCK_LIST;

```

[illegible]

```

BUILD_INPUT_POOL: PROCEDURE FIXED;
  DECLARE (TOPADR, STPNT, VARCONIND, ACTADR, CONS, VALUE, VALPNT,
          CNT, OPTYP, OPNT, NAMPNT) FIXED;

```

```

/*                                     */
/* ADV STATE POINTER TWO              */
/* LOCATIONS                          */
/*                                     */

```







266





```

4579 PERFORM_MEET_OP: PROCEDURE(BLKADR) BIT(1);
4580 DECLARE CUR_STATE_CHGD BIT(1),
4581
4582 TOT_CLASS_CNT FIXED,
4583
4584 SIMP_VARIABLES_PRES FIXED,
4585 MAX_ELEMENTS LITERALLY '90',
4586 OCCTOP FIXED,
4587
4588 OCCNUM (MAX_ELEMENTS) BIT(16),
4589
4590 OCCVAL (MAX_ELEMENTS) BIT(16),
4591 MAX_REFINEMENTS LITERALLY '90',
4592 RTOP FIXED,
4593
4594 REFNVAL (MAX_REFINEMENTS) BIT(16),
4595 REFCVAL (MAX_REFINEMENTS) BIT(16),
4596 REFIVAL (MAX_REFINEMENTS) BIT(16),
4597
4598
4599
4600
4601
4602
4603
4604
4605
4606
4607
4608
4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4620
4621
4622
4623
4624
4625
4626
4627

```

/\* TRUE IF STATE HAS BEEN  
 /\* REDUCED.  
 /\* HOLDS TOTAL NUMBER OF  
 /\* ENTRIES IN CURRENT  
 /\* OPTIMIZED POOL  
 /\* IF SIMP VARIABLES PRESENT  
 /\* MAX SIZE OF OCCURANCE LIST  
 /\* HOLDS NUMBER OF CLASSES IN  
 /\* THE OCCURANCE LIST  
 /\* HOLDS # OF ELEMENTS IN  
 /\* EACH CLASS  
 /\* HOLDS CLASS VALUE#  
 /\* MAX SIZE OF REFINEMENT  
 /\* TOP OF THE REFINEMENT LIST  
 /\* HOLDS REFINED VALUE NUMBER  
 /\* HOLDS CURRENT POOL VALUE#  
 /\* HOLDS VALUE NUMBER IN  
 /\* INPUT POOL  
 /\* CLASS MAPPINGS V(V1, V2)  
 /\* WHERE V1 = VAL# IN  
 /\* CURRENT POOL  
 /\* V2 = VAL# IN INPUT  
 /\* POOL  
 /\* V = NEW CLASS #  
 /\* NEW CLASS NUMBER FOR AN  
 /\* EXPRESSION OPERAND  
 /\* OPERAND TO SEARCH FOR IN  
 /\* THE INPUT EXPRESSION  
 /\* CURRENT EXP OPERANDS  
 /\* COUNTER FOR EXPRESSION  
 /\* CLASS INTERSECTION TABLE  
 /\* POINTER FOR EXPRESSION  
 /\* CLASS INTERSECTION TABLE  
 /\* COUNTER FOR THE NUMBER OF  
 /\* CLASSES IN THE MERGED  
 /\* STATE  
 /\* POINTER FOR THE NEW STATE  
 /\* CURRENT STATE POINTER  
 /\* LOOP CONTROL FOR CURRENT  
 /\* ADR OF CURRENT CLASS  
 /\* ADR OF THE INPUT POOL EXP  
 /\* BEING PROCESSED  
 /\* INDICATOR AS TO TYPE OF  
 /\* CURRENT POOL ENTRY  
 /\* 0 = EXPRESSION  
 /\* 1 = 0 = SIMPLE VARIABLE







[illegible]





[illegible]





[illegible]













```

5159 SET_SEARCH_LIST: PROCEDURE BIT(1);
5160 DECLARE (ANS, TRYFLG, FIND, OVFLOW, COMFLG) BIT(1),
5161 (SPOINT, CNT, OCNT, OPADR, LSONE) FIXED;
5162 FIRSTCALL = TRUE;
5163 COMFLG = TRUE;
5164 ANS = FALSE;
5165 STOP = 0;
5166 SPOINT = 1;
5167
5168 /*
5169 /* PROCESS UNTIL ALL
5170 /* PERMUTATIONS IN CURRENT
5171 /* POOL HAVE BEEN SEARCHED
5172 /* FOR.
5173 /*
5174 DO WHILE COMFLG;
5175 TRYFLG = TRUE;
5176 FIRSTCALLR = TRUE;
5177 FIND = FALSE;
5178
5179 /*
5180 /* PROCESS UNTIL A MATCH IS
5181 /* FOUND OR ALL PERMUTATIONS
5182 /* OF REFINEMENT LIST HAVE
5183 /* BEEN CHECKED.
5184 /*
5185 DO WHILE TRYFLG & ~ FIND;
5186 OVFLOW = FALSE;
5187 CNT = 1;
5188 OCNT = 1;
5189
5190 /*
5191 /* PROCESS A REFINEMENT LIST
5192 /* CONFIGURATION.
5193 /*
5194 DO WHILE (~FIND) & (~OVFLOW);
5195 IF REFCVAL(CNT) = OPERS(OCNT) THEN
5196 DO;
5197 REFOBERS(SPOINT) = REFNVAL(CNT);
5198 INPOBERS(SPOINT) = REFIVAL(CNT);
5199 SPOINT = SPOINT + 1;
5200 FIND = OCNT = NUMGP;
5201 CNT = CNT + 1;
5202 OCNT = OCNT + 1;
5203 LSONE = OCNT - 1;
5204 OVFLOW = (CNT + NUMGP - LSONE) >= RTOP;
5205 END;
5206 ELSE DO;
5207 CNT = CNT + 1;
5208 LSONE = OCNT - 1;
5209 OVFLOW = (CNT + NUMGP - LSONE) >= RTOP;
5210 END;
5211 IF OVFLOW THEN
5212 SPOINT = SPOINT - LSONE;
5213
5214 /*
5215 /* IF CURRENT POOL EXPRESSION
5216 /* FOUND, INDICATE SO.
5217 /*
5218 IF FIND THEN
5219 DO;
5220 ANS = TRUE;
5221 STOP = STOP + 1;
5222 END; ELSE
5223 TRYFLG = PERMUTE_REFINEMENT;
5224 END;
5225
5226 /*
5227 /* IF CURRENT POOL COMMUNITIV
5228 /* TRY TO PERMUTE.
5229 /*
5230 IF COMMOP THEN
5231 DO;
5232 OPADR = ADDR(OPERS(1));
5233 COMFLG = PERMUTE(NUMOP, OPADR, 2, TRUE);
5234 END; ELSE
5235 COMFLG = FALSE;
5236 END;
5237 RETURN ANS;
5238 END SET_SEARCH_LIST;
5239
5240 /*
5241 /* ACTUAL BEGINNING OF PERFORM_MEET_OP.
5242 /*
5243 /*
5244 /*
5245 /* SET THE CURRENT STATE
5246 /* COUNTER AND POINTER
5247 /*
5248 CALL INITIAL_PMO;
5249 IF TRACETOGL THEN

```





```

5244 UU; CALL LINE_FEED(1);
5248 OUTPUT = ' BEFORE THE MEET OPERATION: ';
5249 IF PUNCHTOG THEN
5250 OUTPUT(2) = ' BEFORE THE MEET OPERATION: ';
5251 CALL LINE_FEED(1);
5252 CALL PRINT_OUTPUT_POOL;
5253 CALL W_STATE(BLKADR, 'CURRENT POOL FOR');
5254 END;
5255
5256 /* IS THERE ENOUGH ROOM TO */
5257 /* PERFORM THE MEET OPERATION */
5258 /*
5259
5260 IF (STLOC + STCOUNTER) > MAXSTATE THEN
5261 DO; /* NO... */
5262 CALL ERROR(' UNABLE TO PERFORM MEETOP - STATE TABLE OVERFLOW', 1);
5263 RETURN FALSE;
5264 END;
5265
5266 /* SET POINTERS TO THE MERGED */
5267 /* STATE AREA */
5268 /*
5269 NEW_ST_COUNTER = MAXSTATE - STCOUNTER - 1;
5270 NEW_ST_PT = MAXSTATE - STCOUNTER - 1;
5271 STATEST(NEW_ST_COUNTER) = 0;
5272
5273 /* BUILD THE CLASS # */
5274 /* OCCURANCE LIST FOR THE */
5275 /* CURRENT POOL */
5276 /*
5277 CALL BUILD_CLASS_OCCURANCE_LIST;
5278 IF TABLETOG THEN
5279 CALL W_OCCURRENCE;
5280
5281 /* PROCESS ALL THE SIMPLE */
5282 /* VARIABLES FROM THE CURRENT */
5283 /* STATE */
5284 /*
5285 /* ARE THERE ANY SIMPLE */
5286 /* VARIABLES PRESENT. */
5287 /*
5288 DO WHILE (STCOUNTER > 0) & (SIMP_VARIABLES_PRES /= 0);
5289 /*
5290 /* YES...
5291 /*
5292 STPT = STPT + 1;
5293 STCOUNTER = STCOUNTER - 1;
5294 OPTIMADR = STATES(STPT);
5295 OPTIMVAR = ADDRESS(OPTIMADR);
5296 OPTIMVAL = ADDVAL(OPTIMADR);
5297 OPTITYP = ADDTYPE(OPTIMADR);
5298 IF OPTIMVAR /= 0 THEN
5299 /* IS THIS A SIMPLE VARIABLE */
5300 /* OR AN EXEC STACK ENTRY */
5301 DO;
5302 /*
5303 /* YES...IS IT EXEC STACK */
5304 /*
5305 IF SHR(OPTIMVAR, 15) & "1" THEN
5306 /*
5307 /* YES...GET THE IDENTIFIER */
5308 /*
5309 DO; OPTIMVAR = ADDCON(OPTIMADR) & "7FFF";
5310 CURESTACK = TRUE;
5311 END; ELSE
5312 CURESTACK = FALSE;
5313
5314 /* DECREMENT THE VALUE # */
5315 /* COUNT FOR THIS IDENTIFIER */
5316 /*
5317 CALL DELETE_OCCURANCE(OPTIMVAL);
5318 SIMP_VARIABLES_PRES = SIMP_VARIABLES_PRES - 1;
5319
5320 /* IS THE IDENTIFIER IN THE */
5321 /* INPUT POOL */
5322 /*
5323 INPUTADR = LOOK_VARIABLE(OPTIMVAR);
5324 IF INPUTADR /= 0 THEN
5325 DO;
5326 /*
5327 /* YES...
5328 /*
5329 INPTYP = ADDTYPE(INPUTADR);
5330 INPUTVAL = ADDVAL(INPUTADR);
5331 REFAADR = SEAKCH_REFINEMENT_LIST(OPTIMVAL, INPUTVAL);
5332
5333 /* IS THE ENTRY ALREADY IN */
5334 /*

```



```

5335                                     /* THE REFINEMENT LIST      */
5336                                     /*                               */
5337                                     /*                               */
5338 IF REFAOR = 0 THEN                                     /* NO...PUT IT IN          */
5339                                     /*                               */
5340 REFAOR = ASSIGN_REFINEMENT(OPTIMVAL, INPUTVAL, OPTTYP, INPTYP);
5341                                     /* BUILD A STATE ENTRY FOR */
5342                                     /* THE SIMPLE VARIABLE    */
5343                                     /*                               */
5344 CALL BUILD_NEW_SIM_VAR_ENTRY(REFAOR, OPTIMADR, INPUTADR);
5345                                     /*                               */
5346 END; ELSE                                     /* END OF INPUTADR != 0    */
5347                                     /*                               */
5348                                     /* VARIABLE NOT IN INPUT POOL */
5349                                     /* I.E., CURRENT STATE IS   */
5350                                     /* GOING TO GET SMALLER     */
5351                                     /*                               */
5352 CUR_STATE_CHGD = TRUE;
5353                                     /* END OF SIMPLE VARIABLE */
5354 END;
5355 IF TABLETOG THEN
5356 DO;
5357   OUTPUT = ' AFTER PROCESSING SIMPLE VARIABLES: ';
5358   IF PUNCHTOG THEN
5359     OUTPUT(2) = ' AFTER PROCESSING SIMPLE VARIABLES: ';
5360   CALL W_OCCURRENCE;
5361   CALL W_REFINEMENT;
5362 END;
5363                                     /* NOW PROCESS ALL EXPRESSION */
5364                                     /* FROM CURRENT STATE        */
5365                                     /*                               */
5366 DO WHILE TOT_CLASS_CNT > 0;
5367   STCOUNTER = SAVCNT;
5368   STPT = SAVST;
5369                                     /* PROCESSES ALL CLASSES IN */
5370                                     /* CURRENT POOL.            */
5371                                     /*                               */
5372 DO WHILE STCOUNTER > 0;
5373   STPT = STPT + 1;
5374   STCOUNTER = STCOUNTER - 1;
5375   OPTIMADR = STATES(STPT);
5376                                     /* IS CLASS PRESENT          */
5377                                     /*                               */
5378 IF OPTIMADR != 0 THEN
5379 DO;
5380                                     /* YES...SET INDICATOR      */
5381                                     /*                               */
5382 OPTIMVAR = ADDRESS(OPTIMADR);
5383 OPTIMVAL = ADDVAL(OPTIMADR);
5384 OPTTYP = ADDTYPE(OPTIMADR);
5385                                     /* IS THIS AN EXPRESSION   */
5386                                     /* ENTRY                   */
5387                                     /*                               */
5388 IF OPTIMVAR = 0 THEN
5389 DO;
5390                                     /* YES...                  */
5391                                     /*                               */
5392 EXPADR = ADDNAM(OPTIMADR);
5393 OPER, TEMP = OPTTYP;
5394 COMPOP = OP_TYPE(OPER) = COMM;
5395 NUMOP = OP_DEGL(OPER);
5396 FOUND = FALSE;
5397                                     /* CHECK TO SEE THAT ALL THE */
5398                                     /* OPERANDS HAVE BEEN REMOVED */
5399                                     /* FROM THE CLASS OCCURANCE  */
5400                                     /* LIST                        */
5401                                     /*                               */
5402 I = 1;
5403 DO WHILE NUMOP > 0 & ~ FOUND;
5404   OPVAL = VTAB(EXPADR);
5405   OPERS(I) = OPVAL;
5406   OCCADR = SRCH_OCCURRENCE_LIST(OPVAL);
5407   IF OCCADR != 0 THEN
5408     IF OPVAL == OPTIMVAL THEN
5409       FOUND = TRUE;
5410     I = I + 1;
5411     EXPADR = EXPADR - 1;
5412     NUMOP = NUMOP - 1;
5413   END;
5414 END;
5415                                     /* WAS OPERAND CHECK OK    */

```



```

3423 NUMOP = OP_DEGL(OPER);
3424 IF NOT FOUND THEN
3425 DO;
3426
3427 /* YES...PROCESS THE EXPRESSION */
3428 /* ARE THERE ANY EQUIVALENCE */
3429 /* CLASSES TO TEST FOR */
3430 CALL DELETE_OCCURANCE(OPTIMVAL);
3431 EXPADR = ADDNAM(OPTIMADR);
3432 CURES = OPTIMVAL;
3433
3434 IF SET_SEARCH_LIST THEN
3435 DO;
3436
3437 /* YES...CHECK ALL EXPRESSION */
3438 /* TABLE ENTRIES VIA THE HASH */
3439 /* CODED CHAIN. */
3440
3441 SPOINT = 1;
3442 SVPNT = SPOINT;
3443 DO WHILE (STOP > 0) & (NOT FOUND);
3444 HASHADR = OPER;
3445 DO WHILE NUMOP > 0;
3446 HASHADR = INPOERS(SPOINT) + HASHADR;
3447 SPOINT = SPOINT + 1;
3448 NUMOP = NUMOP - 1;
3449 END;
3450 NUMOP = OP_DEGL(OPER);
3451 SPOINT = SVPNT;
3452 HASHADR = (HASHADR) MOD HASHBASE;
3453 HASHADR = VTAB(HASHADR);
3454 DO WHILE (HASHADR NOT= 0) & (NOT FOUND);
3455 INPUTADR = 0;
3456 I = ADDLOC;
3457 DO WHILE (I > 0) & (INPUTADR = 0);
3458 IF ADDRESS(I) = 0 THEN
3459 IF ADDNAM(I) = HASHADR THEN
3460 INPUTADR = I;
3461 I = I - 1;
3462 END;
3463 IF INPUTADR NOT= 0 THEN
3464 DO;
3465 TSTOPER = VTAB(HASHADR);
3466 TSTOPER = TSTOPER & "FF";
3467 HASHADR = HASHADR + 2;
3468 IF TSTOPER = OPER THEN
3469 DO;
3470 FIND = TRUE;
3471 I = 0;
3472 DO WHILE NUMOP > 0;
3473 IF VTAB(HASHADR + 1) NOT= INPOERS
3474 (SPOINT) THEN
3475 FIND = FALSE;
3476 I = I + 1;
3477 SPOINT = SPOINT + 1;
3478 NUMOP = NUMOP - 1;
3479 END;
3480 NUMOP = OP_DEGL(OPER);
3481 SPOINT = SVPNT;
3482 IF FIND THEN
3483 DO;
3484
3485 /* AN INPUT EXPRESSION HAS */
3486 /* BEEN MATCHED. */
3487
3488 INPTYP = OPTTYP;
3489 INRES = VTAB(HASHADR + NUMOP);
3490 REFADR = SEARCH_REFINEMENT_LIST
3491 (CURES, INRES);
3492 IF REFADR = 0 THEN
3493 REFADR = ASSIGN_REFINEMENT(CURES,
3494 INRES, OPTTYP, INPTYP);
3495 CALL BUILD_EXP_ENTRY(OPTIMADR,
3496 REFADR, INPUTADR);
3497 FOUND = TRUE;
3498 END;
3499 END;
3500
3501 /* ADVANCE TO NEXT HASH CHAIN */
3502 /* ENTRY. */
3503
3504 HASHADR = HASHADR - 1;
3505 END;
3506 ELSE
3507 HASHADR = HASHADR + 1;
3508 HASHADR = VTAB(HASHADR);

```















[illegible]



```

5115 CALL BASIC_BLOCK;
5176
5177
5178
5179
5180
5181 K = ETOP;
5182 DO WHILE ETOP > 0;
5183     CALL ENTERADD(SHL(1, 15) | (K - ETOP));
5184     ADDTYPE(ADDLOC) = EXTYPE(ETOP);
5185     ADDVAL(ADDLOC) = EXVAL#(ETOP);
5186     ADDNAM(ADDLOC) = EXNAM#(ETOP);
5187     I = EXCON(ETOP);
5188     ADDCON(ADDLOC) = SHL(1, 15) | EXADD(ETOP);
5189     ETOP = ETOP - 1;
5190 END;
5191
5192
5193
5194
5195
5196
5197
5198
5199
5200 K = VTAB(VTOP);
5201 DO WHILE K >= 0;
5202     C = VTAB(K + 1) & "FF";
5203     W = OP_DEGL(C) + K + 3;
5204     W = VTAB(W);
5205     DO;
5206         CALL ENTERADD(0);
5207         ADDTYPE(ADDLOC) = C;
5208         ADDVAL(ADDLOC) = W;
5209         ADDNAM(ADDLOC) = K + 1;
5210         ADDCON(ADDLOC) = 0;
5211     END;
5212     K = VTAB(K);
5213 END;
5214
5215
5216
5217
5218
5219 IF PASS# = 1 THEN
5220     CSFCODE(CURBLOCK + 1) = (CSFCODE(CURBLOCK + 1) & "FFFF0000") |
5221         M_CONTROL;
5222
5223
5224
5225
5226
5227
5228
5229
5230
5231
5232
5233
5234
5235
5236 K = #XITS;
5237 K = CTOP - K;
5238 DO WHILE K < CTOP;
5239     K = K + 1;
5240
5241
5242
5243
5244
5245
5246
5247
5248
5249 C = W & "7FFF";
5250 IF SHR(W, 15) & "1" THEN
5251     DO;
5252         IF PERFORM_MEET_OP(C) THEN
5253             DO B = 1 TO K - 1;
5254
5255
5256
5257
5258
5259
5260
5261
5262

```





```

5863
5864
5865
5866
5867
5868
5869
5870
5871
5872
5873
5874
5875
5876
5877
5878
5879
5880
5881
5882
5883
5884
5885
5886
5887
5888
5889
5890
5891
5892
5893
5894
5895
5896
5897
5898
5899
5900
5901
5902
5903
5904
5905
5906
5907
5908
5909
5910
5911
5912
5913
5914
5915
5916
5917
5918
5919
5920
5921
5922
5923
5924
5925
5926
5927
5928
5929
5930
5931
5932
5933
5934
5935
5936
5937
5938
5939
5940
5941
5942
5943
5944
5945
5946
5947
5948
5949
5950

CONTROL(B) = SHL(1, 31);
END; ELSE

CONTROL(K) = SHL(1, 31);
END; ELSE

CALL BUILD_BLOCK(I);
END;

CALL ORDER_BLOCK_LIST;
IF TABLEOG THEN
CALL W_ALL;

END;

END CONTROL_FLOW;

INITIAL_CSF: PROCEDURE;
DECEARE I FIXED;
BLANKS='';

COLLEC=1;
LOAD = 2;
STORE = 3;
STORET = 4;
DELETE = 5;
DUPLIC = 6;
CONV = 7;
COMM = 8;
EXCH = 9;
COND = 10;
UCOND = 11;
DO I = 0 TO NTPES;
NULL(I) = I;
END;
DO I = 1 TO NOPCODES;
REFER(I - 1) = I + NTPES;
END;
GOSUB = 12;
RETSUB = 13;
ETOP = 0;
VTOP = HASHBASE;

DO I = 0 TO HASHBASE;
VTAB (I) = 0;
END;

TRACETOG = 0;
TABLETOG = 0;
PROPTOG = 1;
BRANCHTOG = 1;
PUNCHTOG = 0;

CONLOC, ADDLOC, STLOC = 0;
ADDLOC = MAXADD + 1;
ETOPR = MAXEXEC+1;

/* YES...MARK EARLIER ENTRIES */
/* TO PREVENT PROCESSING */
/*
/* OPTIMIZED POOL WAS NOT
/* ALTERED DURING MEET - DO
/* NOT PROCESS THIS BLOCK
/*
/* BLOCK NOT BUILT
/* BUILD A HEADER FOR IT
/* FWA IS 1 IN CODE
/*
/* WHILE K < CTOP
/* ORDER THE BLOCK SELECTION
/* LIST AS REQUESTED.
/*
/* END OF BLOCK PROCESSED
/*
/* WHILE CTOP > 0
/*
/* INITIAL_CSF
/* ROUTINE IS THE BASIC INITIALIZATION
/* LOOP FOR THE PROGRAM.
/*
/* SET BASIC OPERATIONS TYPES
/*
/* CLEAR HASH ADDRESSES
/*
/* SET TOGGLES
/*
/* SET ALL TABLE POINTERS
/*

```





```

59521 VIOPK=MAXVAL+1;
59522
59523
59524
59525 ALTST = 0;
59526 ALTEX = 0;
59527 ALTAD = 0;
59528 ALTCN = 0;
59529 ALTCT = 0;
59530 ALTVL = 0;
59531 PAGE_EJECT;
59532 OUTPUT = 1;
59533 IF PUNCHTOG THEN
59534 OUTPUT (2) = '
59535 CALL LINE_FEED(2);
59536 END INITIAL_CSF;
59537
59538
59539
59540
59541
59542
59543
59544
59545
59546
59547
59548
59549
59550
59551
59552
59553
59554
59555
59556
59557
59558
59559
59560
59561
59562
59563
59564
59565
59566
59567
59568
59569
59570
59571
59572
59573
59574
59575
59576
59577
59578
59579
59580
59581
59582
59583
59584
59585
59586
59587
59588
59589
59590
59591
59592
59593
59594
59595
59596
59597
59598
59599
59600
59601
59602
59603
59604
59605
59606
59607
59608
59609
59610
59611
59612
59613
59614
59615
59616
59617
59618
59619
59620
59621
59622
59623
59624
59625
59626
59627
59628
59629
59630
59631
59632
59633
59634
59635
59636
59637
59638
59639
59640
59641
59642
59643
59644
59645
59646
59647
59648
59649
59650
59651
59652
59653
59654
59655
59656
59657
59658
59659
59660
59661
59662
59663
59664
59665
59666
59667
59668
59669
59670
59671
59672
59673
59674
59675
59676
59677
59678
59679
59680
59681
59682
59683
59684
59685
59686
59687
59688
59689
59690
59691
59692
59693
59694
59695
59696
59697
59698
59699
59700
59701
59702
59703
59704
59705
59706
59707
59708
59709
59710
59711
59712
59713
59714
59715
59716
59717
59718
59719
59720
59721
59722
59723
59724
59725
59726
59727
59728
59729
59730
59731
59732
59733
59734
59735
59736
59737
59738
59739
59740
59741
59742
59743
59744
59745
59746
59747
59748
59749
59750
59751
59752
59753
59754
59755
59756
59757
59758
59759
59760
59761
59762
59763
59764
59765
59766
59767
59768
59769
59770
59771
59772
59773
59774
59775
59776
59777
59778
59779
59780
59781
59782
59783
59784
59785
59786
59787
59788
59789
59790
59791
59792
59793
59794
59795
59796
59797
59798
59799
59800
59801
59802
59803
59804
59805
59806
59807
59808
59809
59810
59811
59812
59813
59814
59815
59816
59817
59818
59819
59820
59821
59822
59823
59824
59825
59826
59827
59828
59829
59830
59831
59832
59833
59834
59835
59836
59837
59838
59839
59840
59841
59842
59843
59844
59845
59846
59847
59848
59849
59850
59851
59852
59853
59854
59855
59856
59857
59858
59859
59860
59861
59862
59863
59864
59865
59866
59867
59868
59869
59870
59871
59872
59873
59874
59875
59876
59877
59878
59879
59880
59881
59882
59883
59884
59885
59886
59887
59888
59889
59890
59891
59892
59893
59894
59895
59896
59897
59898
59899
59900
59901
59902
59903
59904
59905
59906
59907
59908
59909
59910
59911
59912
59913
59914
59915
59916
59917
59918
59919
59920
59921
59922
59923
59924
59925
59926
59927
59928
59929
59930
59931
59932
59933
59934
59935
59936
59937
59938
59939
59940
59941
59942
59943
59944
59945
59946
59947
59948
59949
59950
59951
59952
59953
59954
59955
59956
59957
59958
59959
59960
59961
59962
59963
59964
59965
59966
59967
59968
59969
59970
59971
59972
59973
59974
59975
59976
59977
59978
59979
59980
59981
59982
59983
59984
59985
59986
59987
59988
59989
59990
59991
59992
59993
59994
59995
59996
59997
59998
59999
60000
60001
60002
60003
60004
60005
60006
60007
60008
60009
60010
60011
60012
60013
60014
60015
60016
60017
60018
60019
60020
60021
60022
60023
60024
60025
60026
60027
60028
60029
60030
60031
60032
60033
60034
60035
60036
60037
60038
60039
60040
60041
60042
60043
60044
60045
60046
60047
60048
60049
60050
60051
60052
60053
60054
60055
60056
60057
60058
60059
60060
60061
60062
60063
60064
60065
60066
60067
60068
60069
60070
60071
60072
60073
60074
60075
60076
60077
60078
60079
60080
60081
60082
60083
60084
60085
60086
60087
60088
60089
60090
60091
60092
60093
60094
60095
60096
60097
60098
60099
60100
60101
60102
60103
60104
60105
60106
60107
60108
60109
60110
60111
60112
60113
60114
60115
60116
60117
60118
60119
60120
60121
60122
60123
60124
60125
60126
60127
60128
60129
60130
60131
60132
60133
60134
60135
60136
60137
60138
60139
60140
60141
60142
60143
60144
60145
60146
60147
60148
60149
60150
60151
60152
60153
60154
60155
60156
60157
60158
60159
60160
60161
60162
60163
60164
60165
60166
60167
60168
60169
60170
60171
60172
60173
60174
60175
60176
60177
60178
60179
60180
60181
60182
60183
60184
60185
60186
60187
60188
60189

```



```

6039 CALL BUILD_BLOCKIN_CONTROL;
6040
6041
6042
6043
6044 END INITIAL_METAEX;
6045
6046
6047
6048
6049
6050
6051
6052
6053
6054
6055
6056
6057
6058
6059
6060
6061
6062
6063
6064
6065
6066
6067
6068
6069
6070
6071
6072
6073
6074
6075
6076
6077
6078
6079
6080
6081
6082
6083
6084
6085
6086
6087
6088
6089
6090
6091
6092
6093
6094
6095
6096
6097
6098
6099
6100
6101
6102
6103
6104
6105
6106
6107
6108
6109
6110
6111
6112
6113
6114
6115
6116
6117
6118
6119
6120
6121
6122
6123
6124
6125
6126
6127
6128
6129
6130
6131
6132
6133
6134
6135
6136
6137
6138
6139
6140
6141
6142
6143
6144
6145
6146
6147
6148
6149
6150
6151
6152
6153
6154
6155
6156
6157
6158
6159
6160
6161
6162
6163
6164
6165
6166
6167
6168
6169
6170
6171
6172
6173
6174
6175
6176
6177
6178
6179
6180
6181
6182
6183
6184
6185
6186
6187
6188
6189
6190
6191
6192
6193
6194
6195
6196
6197
6198
6199
6200
6201
6202
6203
6204
6205
6206
6207
6208
6209
6210
6211
6212
6213
6214
6215
6216
6217
6218
6219
6220
6221
6222
6223
6224
6225
6226
6227
6228
6229
6230
6231
6232
6233
6234
6235
6236
6237
6238
6239
6240
6241
6242
6243
6244
6245
6246
6247
6248
6249
6250
6251
6252
6253
6254
6255
6256
6257
6258
6259
6260
6261
6262
6263
6264
6265
6266
6267
6268
6269
6270
6271
6272
6273
6274
6275
6276
6277
6278
6279
6280
6281
6282
6283
6284
6285
6286
6287
6288
6289
6290
6291
6292
6293
6294
6295
6296
6297
6298
6299
6300
6301
6302
6303
6304
6305
6306
6307
6308
6309
6310
6311
6312
6313
6314
6315
6316
6317
6318
6319
6320
6321
6322
6323
6324
6325
6326
6327
6328
6329
6330
6331
6332
6333
6334
6335
6336
6337
6338
6339
6340
6341
6342
6343
6344
6345
6346
6347
6348
6349
6350
6351
6352
6353
6354
6355
6356
6357
6358
6359
6360
6361
6362
6363
6364
6365
6366
6367
6368
6369
6370
6371
6372
6373
6374
6375
6376
6377
6378
6379
6380
6381
6382
6383
6384
6385
6386
6387
6388
6389
6390
6391
6392
6393
6394
6395
6396
6397
6398
6399
6400
6401
6402
6403
6404
6405
6406
6407
6408
6409
6410
6411
6412
6413
6414
6415
6416
6417
6418
6419
6420
6421
6422
6423
6424
6425
6426
6427
6428
6429
6430
6431
6432
6433
6434
6435
6436
6437
6438
6439
6440
6441
6442
6443
6444
6445
6446
6447
6448
6449
6450
6451
6452
6453
6454
6455
6456
6457
6458
6459
6460
6461
6462
6463
6464
6465
6466
6467
6468
6469
6470
6471
6472
6473
6474
6475
6476
6477
6478
6479
6480
6481
6482
6483
6484
6485
6486
6487
6488
6489
6490
6491
6492
6493
6494
6495
6496
6497
6498
6499
6500
6501
6502
6503
6504
6505
6506
6507
6508
6509
6510
6511
6512
6513
6514
6515
6516
6517
6518
6519
6520
6521
6522
6523
6524
6525
6526
6527
6528
6529
6530
6531
6532
6533
6534
6535
6536
6537
6538
6539
6540
6541
6542
6543
6544
6545
6546
6547
6548
6549
6550
6551
6552
6553
6554
6555
6556
6557
6558
6559
6560
6561
6562
6563
6564
6565
6566
6567
6568
6569
6570
6571
6572
6573
6574
6575
6576
6577
6578
6579
6580
6581
6582
6583
6584
6585
6586
6587
6588
6589
6590
6591
6592
6593
6594
6595
6596
6597
6598
6599
6600
6601
6602
6603
6604
6605
6606
6607
6608
6609
6610
6611
6612
6613
6614
6615
6616
6617
6618
6619
6620
6621
6622
6623
6624
6625
6626
6627
6628
6629
6630
6631
6632
6633
6634
6635
6636
6637
6638
6639
6640
6641
6642
6643
6644
6645
6646
6647
6648
6649
6650
6651
6652
6653
6654
6655
6656
6657
6658
6659
6660
6661
6662
6663
6664
6665
6666
6667
6668
6669
6670
6671
6672
6673
6674
6675
6676
6677
6678
6679
6680
6681
6682
6683
6684
6685
6686
6687
6688
6689
6690
6691
6692
6693
6694
6695
6696
6697
6698
6699
6700
6701
6702
6703
6704
6705
6706
6707
6708
6709
6710
6711
6712
6713
6714
6715
6716
6717
6718
6719
6720
6721
6722
6723
6724
6725
6726
6727
6728
6729
6730
6731
6732
6733
6734
6735
6736
6737
6738
6739
6740
6741
6742
6743
6744
6745
6746
6747
6748
6749
6750
6751
6752
6753
6754
6755
6756
6757
6758
6759
6760
6761
6762
6763
6764
6765
6766
6767
6768
6769
6770
6771
6772
6773
6774
6775
6776
6777
6778
6779
6780
6781
6782
6783
6784
6785
6786
6787
6788
6789
6790
6791
6792
6793
6794
6795
6796
6797
6798
6799
6800
6801
6802
6803
6804
6805
6806
6807
6808
6809
6810
6811
6812
6813
6814
6815
6816
6817
6818
6819
6820
6821
6822
6823
6824
6825
6826
6827
6828
6829
6830
6831
6832
6833
6834
6835
6836
6837
6838
6839
6840
6841
6842
6843
6844
6845
6846
6847
6848
6849
6850
6851
6852
6853
6854
68
```



## BIBLIOGRAPHY

1. Aho, A. V., Sethi, R. and Ullman, J. D., "Code Optimization on Finite Church-Rosser Systems," Design and Optimization of Compilers, p. 69-106, 1972.
2. Aho, A. V. and Ullman, J. D., "Optimization of Straight Line Code," SIAM Journal on Computing, v. 1, n. 1, p. 1-19, March 1972.
3. Aho, A. V. and Ullman, J. D., The Theory of Parsing, Translation, and Compiling Vol. II: Parsing, Prentice-hall, Inc., 1973.
4. Allen, F. E. and Cocke, J., "A Catalogue of Optimizing Transformations," Design and Optimization of Compilers, p. 1-30, 1972.
5. Anderson, J., "A Note on some Compiling Algorithms," Communications of the ACM, v. 7, n. 3, p. 149-150, March 1964.
6. Bagwell Jr., J. T., "Local Optimizations," SIGPLAN Notices of the ACM, v. 5, n. 7, p. 52-66, July 1970.
7. Busam, V. and Englund, D., "Optimization of Expressions in FORTRAN," Communications of the ACM, v. 12, n. 12, p. 666-674, December 1969.
8. Cheatham, T. E. and Standish T., "Optimization Aspects of Compiler-Compilers," SIGPLAN Notices of the ACM, v. 5, n. 10, p. 10-17, October 1970.
9. Cocke, J., "Global Common Subexpression Elimination," SIGPLAN Notices of the ACM, v. 5, n. 7, p. 20-24, July 1970.





10. Cocke, J. and Schartz, J. T., Programming Languages and their Compilers, Preliminary Notes, 2nd ed., p. 306-524, Courant Institute of Mathematical Sciences New York University, 1970.
11. Day, W., "Compiler Assignment of Data Items to Registers," IBM Systems Journal, v. 9, n. 4, p. 281-317, 1970.
12. Earnest, C. P., "Some Topics in Code Optimization," Journal of the ACM, v. 21, n. 1, p. 76-102, January 1974.
13. Elson, M., and Rake, S., "Code Generation Technique for Large Language Compilers," IBM Systems Journal, v. 9, n. 3, p. 166-188, 1970.
14. Fateman, R., "Optimal Code for Serial and Parallel Computation," Communications of the ACM, v. 12, n. 12, p. 694-695, December 1969.
15. Finkelstein, M., "A Compiler Optimization Technique," Computer Review, p. 22-25, February 1968.
16. Floyd, R., "An Algorithm for Coding Efficient Arithmetic Operations," Communications of the ACM, v. 4, n. 1, p. 42-51, January 1961.
17. Fong, A., Kan, J. and Ullman, J., "Application of Lattice Algebra to Loop Optimization," Conference Record of the Second ACM Symposium on Principles of Programming Languages, p. 1-9, Palo Alto, California, January 1975.
18. Frailey, D. J., "Expression Optimization Using Unary Complement Operators," SIGPLAN Notices of the ACM, v. 5, n. 7, p. 67-85, July 1970.





19. Goldberg, P., "A Comparison of Certain Optimization Techniques," Design and Optimization of Compilers, p. 31-50, 1972.
20. Graham, S. L. and Wegman, M., "A Fast and Usually Linear Algorithm for Global Flow Analysis," Conference Record of the Second ACM Symposium on Principles of Programming Languages, p. 22-34, Palo Alto, California, January 1975.
21. Haynes, H. R. and Schutte, L. J., "Compilation of Optimized Syntactic Recognizers from Floyd-Evans Productions," SIGPLAN Notices of the ACM, v. 5, n. 7, p. 38-51, July 1970.
22. Hecht, M. S. and Ullman, J. D., "Analysis of a Simple Algorithm for Global Data Flow Problems," Conference Record of the ACM Symposium on Principles of Programming Languages, p. 207-217, Boston, Massachusetts, October 1973.
23. Hecht, M. S. and Ullman, J. D., "Characterizations of Reducible Flow Graphs," Journal of the ACM, v. 21, n. 3, p. 367-375, July 1974.
24. Hecht, M. S. and Ullman, J. D., "Flow Graph Reducibility," SIAM Journal on Computing, v. 1, n. 2, p. 188-202, June 1972.
25. Kennedy, K., "A Global Flow Analysis Algorithm," International Journal of Computer Mathematics, Section A, v. 3, n. 1, p. 5-15, December 1971.
26. Kennedy, K., "Node Listings Applied to Data Flow Analysis," Conference Record of the Second ACM Symposium on Principles of Programming Languages, p. 10-21, Palo Alto, California, January 1975.



27. Kennedy, K., "Safety of Code Motion," International Journal of Computer Mathematics, Section A, v. 3, n. 2, p. 117-130, September 1972.
28. Kildall, G. A., "A Unified Approach to Global Optimization," Conference Record of the ACM Symposium on Principles of Programming Languages, p. 194-206, Boston, Massachusetts, October 1973.
29. Kildall, G. A., A Code Synthesis "Filter" for Basic Block Optimization, Computer Science Group Report TR #72-3-01, University of Washington, Seattle, January 1972.
30. Kildall, G. A., Global Expression Optimization During Compilation, Ph. D. Dissertation, Computer Science Group, University of Washington, Seattle, June 1972.
31. Kildall, G. A., The ALGOL-E Programming System, Internal Report, Mathematics Department, Naval Postgraduate School, Monterey, California, December 1970.
32. Kildall, G. A. and Roberts, A., "ALGOL-E: An Experimental Approach to the Study of Programming Languages," SIGCSE Bulletin, n. 4, p. 127-135, March 1972.
33. Lowery, E. S. and Medlock, C. W., "Object Code Optimization," Communications of the ACM, v. 12, n. 1, p. 13-22, January 1969.
34. Lukasczyk, N., An Investigation of Selection Methods for a Simple Program Flow Analysis Algorithm, Masters Thesis, U. S. Naval Postgraduate School, Monterey, June 1974.



35. Maggiolo-Schettini, A., "Procedure Linkage Optimization Working Paper," Conference Record of the ACM Symposium on Principles of Programming Languages, p. 183-193, Boston, Massachusetts, October 1973.
36. McKeeman, W., "Peephole Optimization," Communications of the ACM, v. 8, n. 7, p. 443-444, July 1965.
37. McKeeman, W. M., Horning, J. J. and Wortman, D. B., A Compiler Generator, Prentice-Hall, Inc., 1970.
38. Naur, P., "Revised Report on the Algorithmic Language ALGOL 60," Communications of the ACM, v. 6, p. 1-17, January 1963.
39. Painter, J. A., "Effectiveness of an Optimizing Compiler for Arithmetic Expressions," SIGPLAN Notices of the ACM, v. 5, n. 7, p. 101-126, July 1970.
40. Rosen, B. K., "Note on Semantics and Optimization," SIGPLAN Notices of the ACM, v. 9, n. 11, p. 6-10, November 1974.
41. Sethi, R. and Ullman, J. D., "Generation of Optimal Code for Arithmetic Expressions," Journal of the ACM, v. 17, n. 4, p. 715-728 October 1970.
42. Tarjan, R. H., "Depth-First Search and Linear Graph Algorithms," SIAM Journal on Computing, v. 1, n. 2, p. 146-160, June 1972.
43. Urschler, G., Complete Redundant Expression Elimination in Flow Diagrams, IBM Thomas J. Watson Research Center Computer Sciences Department RC 4965 (#22020), August 1974.
44. Zelkowitz, M. V. and Bail, W. G., "Optimization of Structured Programs," Software - Practice and Experience, v. 4, p. 51-57, 1971.



# INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2. Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3. Chairman, Code 72 Computer Science Group Naval Postgraduate School Monterey, California 93940	2
4. Professor Gary A. Kildall, Code 72Kd (advisor) Naval Postgraduate School Monterey, California 93940	1
5. Professor Daniel L. Davis, Code 53Dv Naval Postgraduate School Monterey, California 93940	1
6. Ensign Gerald G. Brown USN, Code 55Zr Naval Postgraduate School Monterey, California 93940	1
7. Lieutenant(JG) Jack W. Cowan USN (student) NAVCOSACT Washington Navy Yard Washington, D. C. 20390	1





160682

Thesis  
C756923  
c.1

Cowan

An implementation  
of an iterative glo-  
val flow analysis  
algorithm.

160582

Thesis  
C756923  
c.1

Cowan

An implementation  
of an iterative glo-  
bal flow analysis  
algorithm.

thesC756923

An implementation of an iterative global



3 2768 002 08996 3

DUDLEY KNOX LIBRARY